AD_____

Award Number: DAMD17-01-1-0828

TITLE: Sixth Phantom Users Group Conference

PRINCIPAL INVESTIGATOR: Karl D. Reinig, Ph.D.

CONTRACTING ORGANIZATION: University of Colorado Health Sciences Center, Fitzsimons
Aurora, Colorado 80045-0508

REPORT DATE: January 2002

TYPE OF REPORT: Final Proceedings

PREPARED FOR: U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

DISTRIBUTION STATEMENT: Approved for Public Release;
Distribution Unlimited

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 074-0188*



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | January 2002 | Final Proceedings (1 Sep 01 - 31 Dec 02) |

**4. TITLE AND SUBTITLE**
Sixth Phantom Users Group Conference

**5. FUNDING NUMBERS**
DAMD17-01-1-0828

**6. AUTHOR(S) :**
Karl D. Reinig, Ph.D.

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
University of Colorado Health
Sciences Center, Fitzsimons
Aurora, Colorado 80045-0508
Email: karl@chs.uchsc.edu

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**



**11. SUPPLEMENTARY NOTES**
report contains color

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for Public Release; Distribution Unlimited

**12b. DISTRIBUTION CODE**

**13. Abstract** ***(Maximum 200 Words)*** *(abstract should contain no proprietary or confidential information)*

This grant represented the major funding for the Sixth PHANToM Users Group Conference. The conference was held in a workshop style at the Given Institute in Aspen Colorado. This was the second time that the University of Colorado hosted the conference. The conference successfully attracted haptic researchers from around the world In addition to the individual papers, the conference included a discussion of haptic standards and multiple tutorials.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| haptics, simulation | 114 |
| | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | Unlimited |

# Table of Contents

**Final Report:** Support for conference entitled "The Sixth PHANToM Users Group Workshop"

**Award No.**: DAMD17-01-1-0828

## *Introduction*

### General

The Center for Human Simulation hosted the Phantom Users's Group Workshop in Aspen Colorado for the second straight year. TATRC's funding was critical to the success of the conference.

The following lists titles of the papers presented at the workshop.

- "The Active Polygon Polygonal Algorithm for Haptic Force Generation" Tom Anderson, Nick Brown
  *Novint Technologies*

- "A Dynamic Design Strategy for Visual and Haptic Development"
  Arthurine Breckenridge, Ben Hamlet
  *Sandia National Laboratories*
  Derek Mehlhorn
  *Univeristy of Washington*
  Kevin Oishi
  *Carnegie Mellon University*

- "Dealing with Desktop Gimbal Noise"
  Karl Reinig, Chris Lee

- "Expanding the Haptic Experience by Using the PHANToM as a Camera Metaphor"
  Joan De Boeck, Chris Raymaekers, Karin Coninx
  Expertise Center for Digital Media
  *Limburg University Center*

- "Scene Complexity: A measure for real-time stable haptic applicaitons"
  Eric Acosta, Bharti Temkin
  Department of Computer Science
  *Texas Tech University*

- "The Sound and Touch of Mathematics: a prototype System"
  Francis L. Van Scoy, Takamitsu Kawai, Angela Fullmer, Kevin Stamper
  *West Virginia University*

Iwona Wojciechowska
*Alderson Broaddus College*
Addis Perez, Jesir Vargas
*University of Puerto Rico - Rio Piedras*
Shelma Martinez
*University of Puerto Rico - Mayaguez*

- "Fast Haptic Rendering of Comlex Objects Using Subdivision Surfaces"
  Chris Raymaekers, Koen Beets, Frank Van Reeth
  *Limburg University Centre*

- "Improving our nomenclature and (why it matters), Ted Kirkpatrick, *Simon Fraser University*

- "Military Medical Modeling & simulation in the 21st Century": An Update, J. Harvey Magee

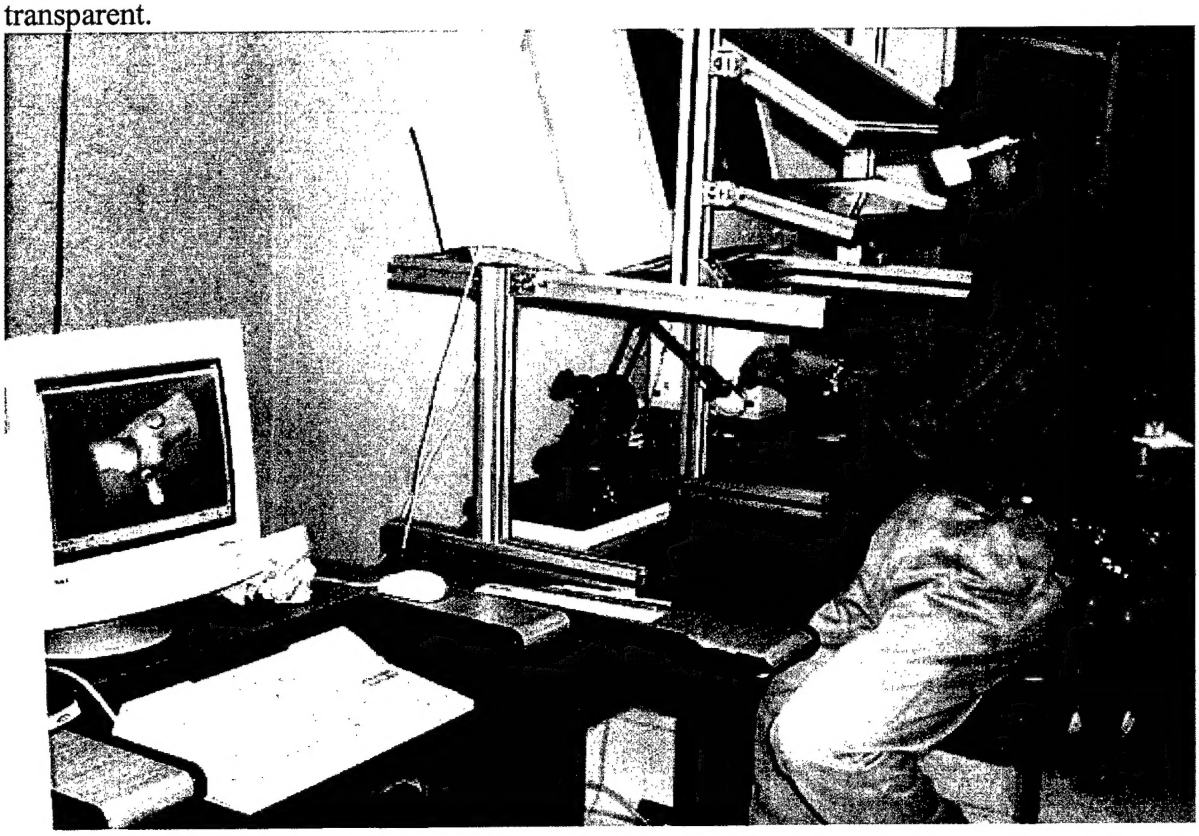
In addition, we held a session to discuss standards for haptic display and GHOST, e-Touch, and freeform tutorials.

## *Body*

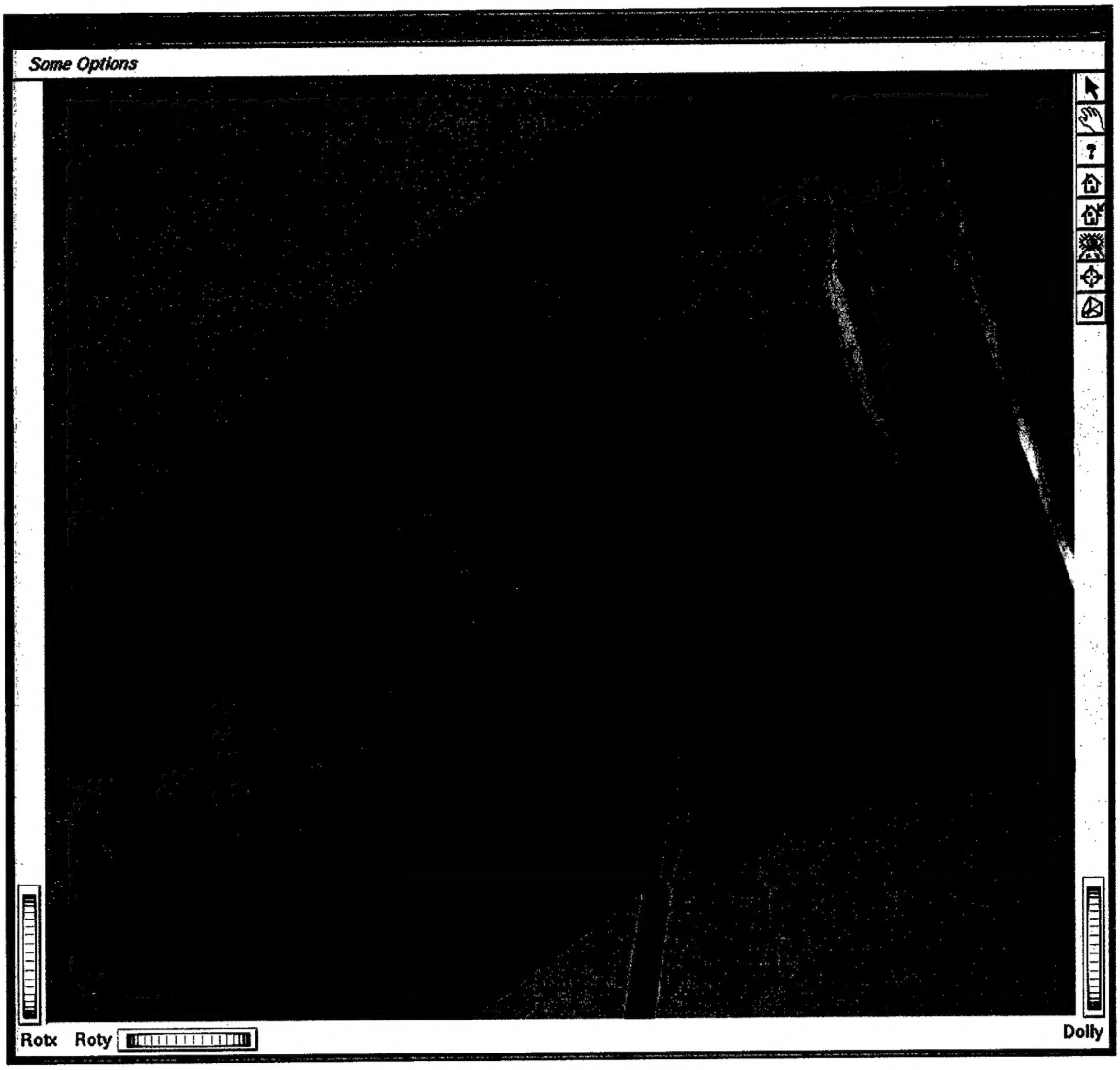
### CHS Open House

PUG 2001 began with an open-house at the University of Colorado's Center for Human Simulation (CHS). The CHS is located on the former Fitzsimons Army Medical Center in Aurora, Colorado.

The open-house included six stations. The first two stations were simulator prototypes. The first of these demonstrated a 3-D knee that could be palpated, cut with a scalpel, or injected or aspirated with a needle/syringe. The workstation is shown below with the injection/aspiration mode selected and all but the bones and ligaments of the knee made transparent.

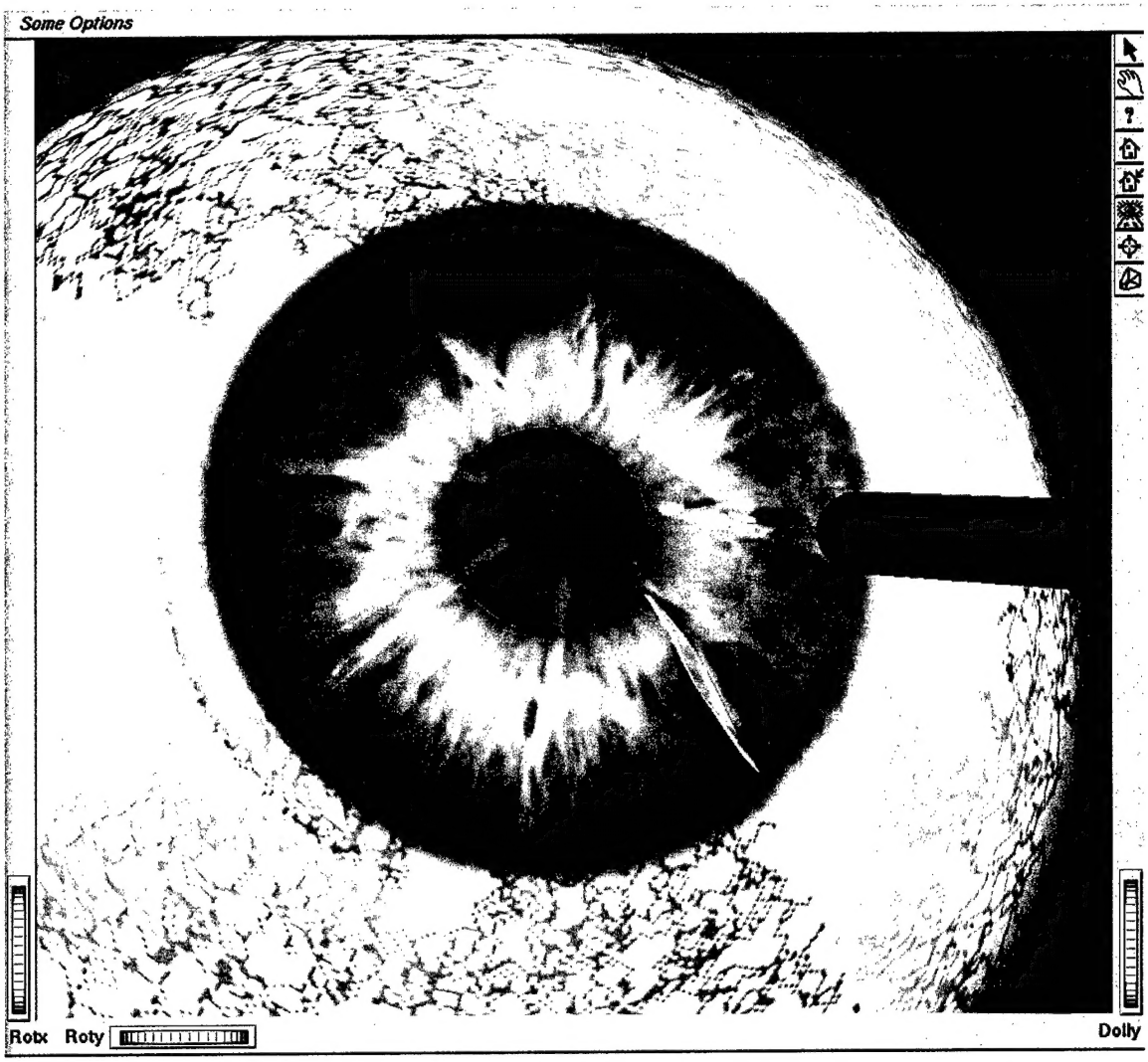


The following figure shows incisions produced by the simulator.




The second simulator prototype allowed the user to practice corneal incisions on a virtual eye. The prototype keeps track of all scalpel motion and tells the user if they have started, finished, or wandered outside of an accepted tolerance during a cut. It also has a testing mode that gives the user a score. The simulator is ergonomically very similar to cutting real tissue under a microscope. We have prototyped a study, that produced encouraging results, to measure skills acquired with this simulator. This prototype was built to demonstrate that we have developed the fundamentals required to create a phaco-emulsification simulator on which to practice cataract removal. We feel that such a

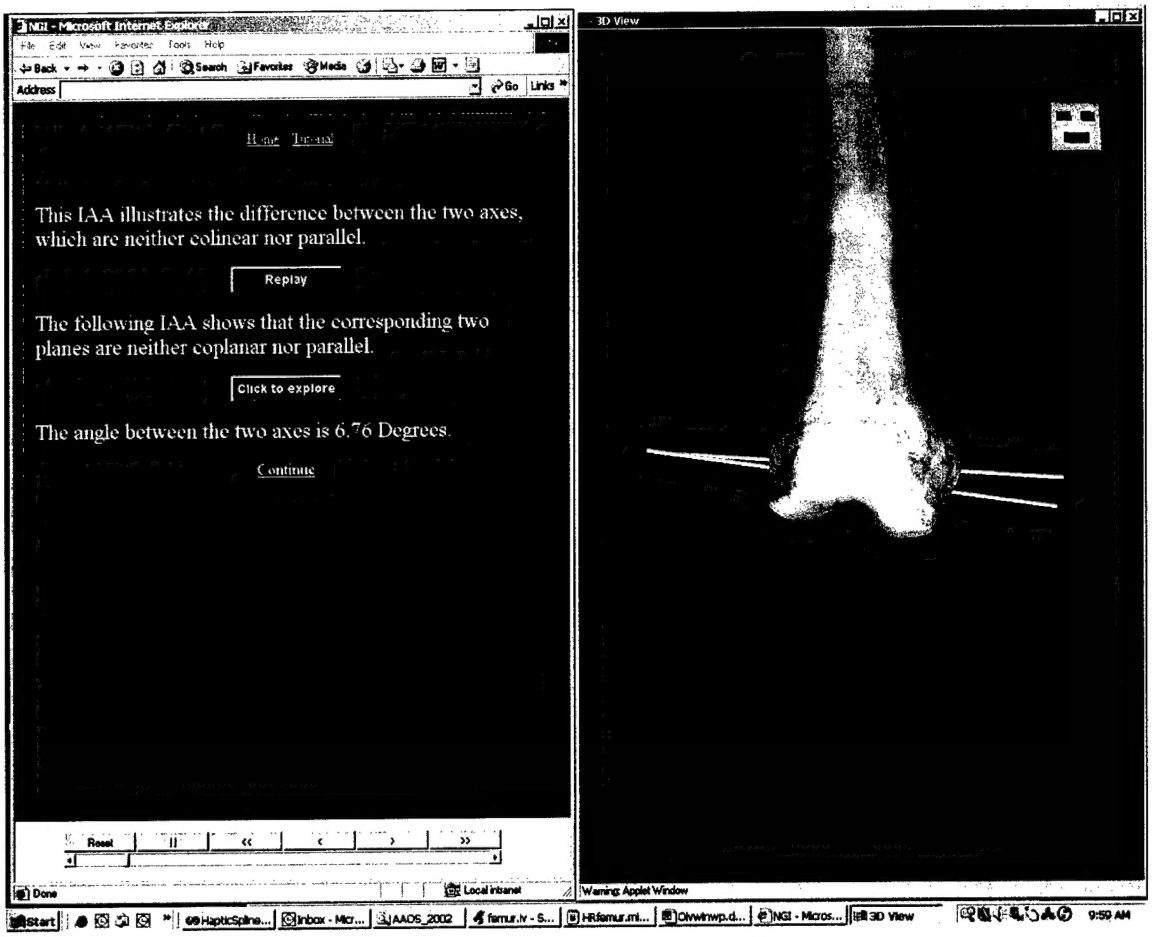
simulator could have a large impact on ophthalmic training, bringing a significant benefit to society.



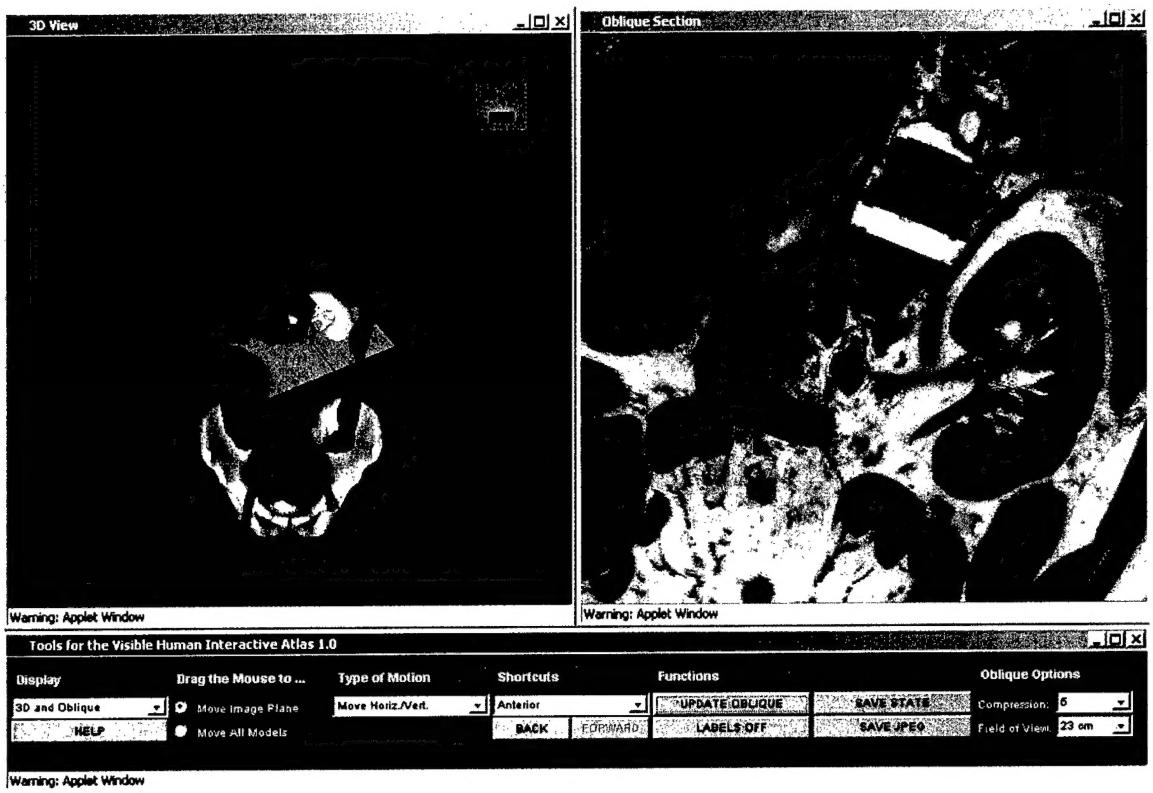

The third station showed the current state of our Explorable Virtual Human (EVH) being developed as part of a Next Generation Internet (NGI) contract with the National Library of Medicine. The EVH is an authoring/display tool for virtual anatomy. The models that we are producing from the Visible Human data have greatly improved the reality of anatomy available for display in virtual environments. Our "Solid Shells" technique gives the models the fidelity of ray-traced models while rendering at VR rates. The EVH will make these models available to a diverse audience of developers.

The following is a screen capture of one page of the EVH as it was demonstrated at PUG 2001. The window on the right is a stereoscopic anatomic display of the knee as it flexes. The window on the left displays tightly coupled HTML. Using the EVH editor, the user

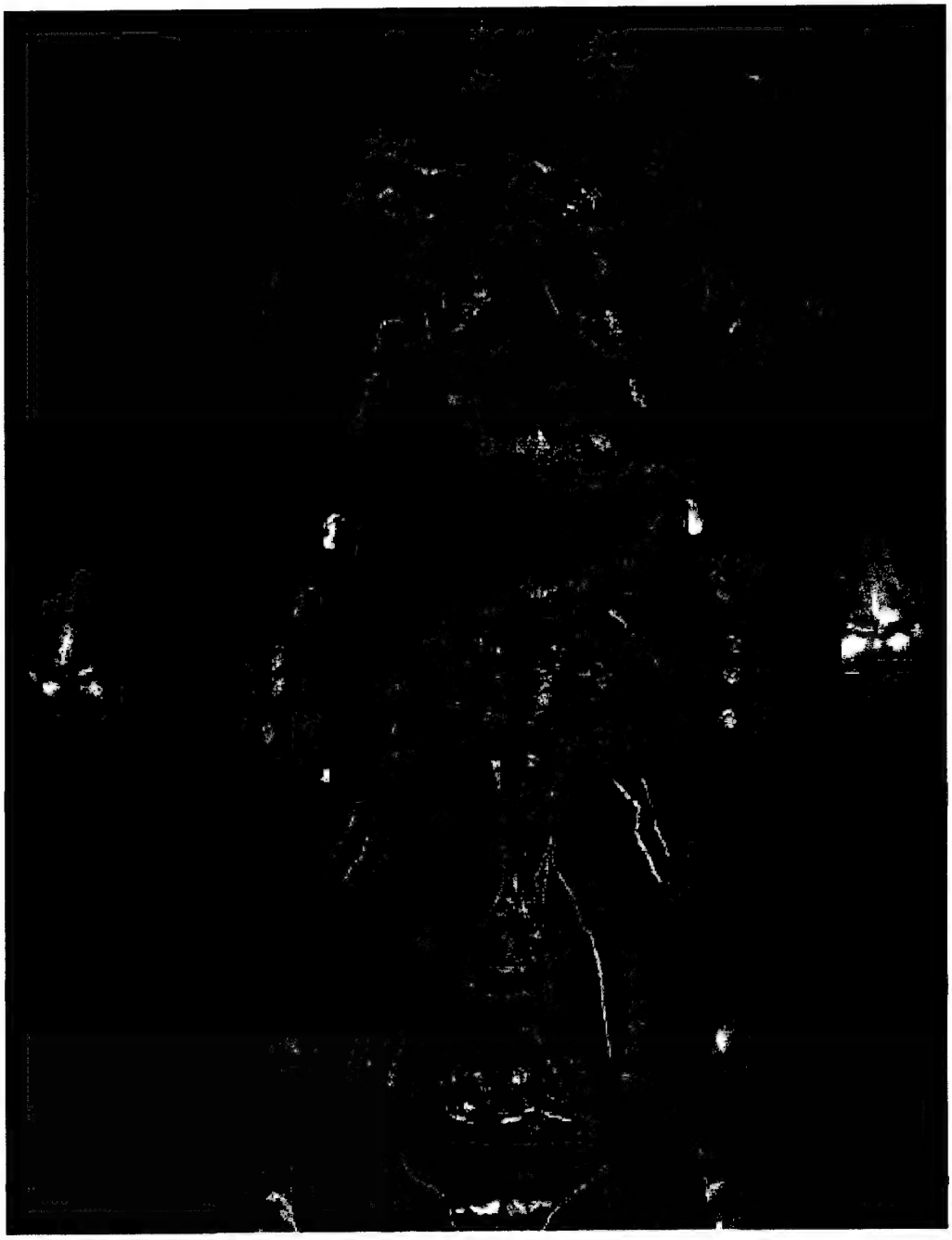
selects from the available models, creates virtual camera motions, and toggles the visibility of structures as a function of time. The resulting Interactive Anatomic Animations (IAAs) are played back in conjunction with the HTML. Whatever anatomic structure the cursor covers identifies itself in the HTML window. Questions may be inserted using the HTML and the results used to determine the next IAA. The image below shows a screen capture of an IAA that was part of a demonstration at the 2001 Annual meeting of the American Academy of Orthopaedic Surgeons. The demonstration garnered their top prize for scientific exhibits.




The fourth station demonstrated an anatomic navigator designed to help teach ultrasound anatomy to GI physicians. The "oblique maker" is an intuitive interface for the production of oblique views of Visible Human type data. The user manipulates a flat plate through 3-D models of their choosing. When they select "Update Oblique" they are given the corresponding oblique slice through the Visible Human Male. Structures in the oblique image identify themselves when selected by the mouse. The oblique maker runs over the net www.visiblehumanexperience.com.




The fifth station used the Center's one-wall cave to demonstrate our "Dissector" program. The dissector allows the user to select structures for identification, then remove them if desired. The dissector presents the results using passive stereo. The following is a screen capture of the dissector in use. Here, the structures anterior to the kidneys have been removed and the liver is currently hi-lighted.



The sixth station was a tour of our cutting area. The Fitzsimon's morgue was given to us in pristine condition. It has significantly improved our ability to create Visible Human type data. One of the data sets we have produced since arriving at Fitzsimons is our $1/10^{th}$ mm knee. The knee has more data in it than the entire Visible Human Male. It is being used in our partnership with the American Academy of Orthopaedic Surgeons to produce an arthroscopy simulator.

### Tutorials

Once again, Authur "Ted" Kirkpatrick (Simon Fraser University ted@sfu.ca) gave a Ghost Tutorial. This carefully planned tutorial used the standard Ghost Tutorial developed by SensAble Technologies as a prerequisite. This had the effect of leveling the field amongst the students and allowed the tutorial to be more advanced than usual. Ted posed problems to the students and then discussed their answers as well as his own. Attachment one is the handout from the class.

Tom Anderson introduced e-touch to the group in the form of a tutorial. He also explained how the open sourcing of the code would work.

SensAble Technologies also gave a freeform tutorial.

## *Key Research Accomplishments*

### Workshop

The main paper sessions ran from 8:45 – 5:30 on the 27th of October and 8:45 – 4:30 on the following day.

### Haptic Demonstrations

Once again, John Ranta (from Novint) brought haptic devices and computers, on loan from SensAble technologies. He set up a separate demo room that ran throughout the workshop. Software for the demos was provided a priori by the attendees. Getting various haptic demonstrations running on machines that are slightly different then where they were developed has always been a pretty big task, but it creates a powerful addition to the workshop. It is one thing to discuss haptic algorithms, it is quite a different thing to share them. John handled the task smoothly.

## *Reportable Outcomes*

The PUG 2001 conference proceedings are available on line at:
http://www.cs.sandia.gov/SEL/conference/pug01/papers.htm

and are included in the appendix. Conference proceedings for PUG 2002 are also included in the appendix.

## *Conclusions*

As haptic display becomes more entrenched in training, it can be expected to have an ever increasing impact on the military. PUG is a unique venue for those developing haptic display. TATRC's funding was critical for maintaining this venue while it transitioned from SensAble Technologies' offspring to a user supported event. While TATRC's support will always be welcome, it is anticipated that next year's (and future) PUGs will survive with moderate corporate sponsorship.

## *Appendix*

PUG 2001 proceedings attached:

PUG 2002 proceedings attached:

Preprints of the
Sixth Annual

# **PHANToM** Users Group Workshop

October 27-30, 2001
Given Institute, Aspen Colorado
University of Colorado Health Science Center, Denver, Colorado

*Hosted by*

Center for Human Simulation, UCHSC, Denver, CO

*Sponsored by*

TATRC
&
SensAble Technologies

*with demo support from*

Novint Technologies

# The ActivePolygon Polygonal Algorithm for Haptic Force Generation

Tom Anderson
Nick Brown
Novint Technologies

## Abstract

Algorithms for computing forces and associated surface deformations from a polygonal data set are given, which can be used to haptically and graphically display virtual objects with a single-point cursor. A culled collision detection algorithm is described that works in real-time with large data sets utilizing an oct-tree method. After a collision is detected, forces are created based on the local area near the cursor, keeping track of an active polygon. This creates a method that is effective and scalable for large models. The 'Bendable Polygon' algorithm for visual rendering of computer generated surfaces is also given.

## Introduction

Many of the most common data formats for 3D computer-generated worlds utilize polygonal representations for objects. In order to take advantage of the large existing quantity of polygonal data sets, and a common standard in environments and hardware acceleration, a method for creating haptics forces based on polygons is necessary. The following paper describes an algorithm that can be used to create the forces on polygonal objects. The ActivePolygon algorithm was implemented with triangular datasets, however the concepts can apply to any type of polygonal dataset.

The algorithm first focuses on determining if the user point, or cursor, has touched an object, which requires a collision detection algorithm. A culled collision detection algorithm is described that works in real-time with large data sets. Then forces are created based on penetration depth and the relative position of the cursor to the object's facets. Graphically, the Bendable Polygon technique is described in which a local area of an object is broken up to allow for small-scale visual deformations. Larger scale deformations occur in the haptic domain through a system of springs and dampers.

## Collision Detection

The first step in creating the forces for an object is to find if the cursor is touching the object. This means that as the cursor moves, collisions between the cursor and the object's facets must be checked. After a collision is detected, forces are then determined and presented to the user.

A simple way to do collision detection is to check if the cursor has moved through any of the polygons in an object. This can be accomplished by taking the line segment from the cursor's current and previous positions each loop of the cycle, and comparing that segment with every one of the polygons in an object. If the line segment intersects any of the polygons then a collision has occurred.

This can be extremely time consuming, however, if the object consists of many polygons. It is inefficient to check every one of the polygons in an object each cycle of the loop. The process can be sped up by pre-processing the data and culling the polygons that are not in the cursor's vicinity during

run-time, which allows real-time collision detection even with large data sets. An oct-tree is used to subdivide the space around the object. During collision detection, each cycle of the haptic loop all leaf nodes which the cursor has moved through are determined (usually this is only the current leaf node) and only the polygons within these nodes are checked for collisions.

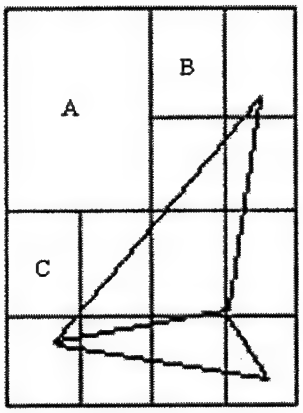



**Figure 1: oct-tree based culling of a polygonal object.**
**Nodes A, B and C are empty, the remaining nodes contain one or both of the polygons.**

To populate the oct-tree, first all the polygons are added to a single parent node. This node is then divided into eight octants all polygons within the parent are checked for overlapping each child octant, and added to the child's polygon list. Then, each of the octants is divided and it's polygon list transferred to the new children. An octant is not subdivided if contains a minimum number of polygons.

In pre-processing the oct-tree culling data, there are three situations in which a polygon should be included in an octant's checking domain as shown in Figure 2. The first is when any vertices of the polygon lie within the octant. The second occurs when any of the edges of the polygon intersect any of the sides of the octant's. The third situation occurs when any of the edges of the octant's intersect the polygon.

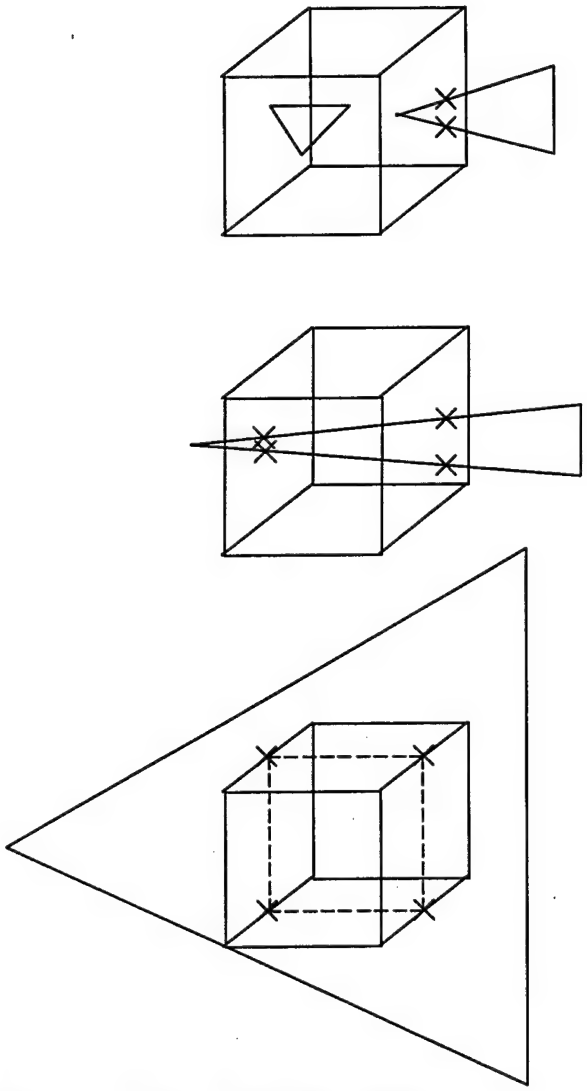


**Figure 2: Voxel-Polygon events.**
**Top: 1 or more vertices in the voxel.**
**Middle: Polygon edge intersects a voxel side.**
**Bottom: Voxel edge intersects the polygon**

If the tool's start position in the X direction is larger than its end position, octants are queried from right to left, otherwise from left to right. Setting the same checks in Y and Z ensures that the octant that will be collided with first is checked first.

The main consideration in using this type of culling comes from trade-offs in time and memory usage. The culling represents a method in which the object's size can grow to contain many millions of polygons, and the algorithm would still be able to do the collision detection in real-time. This would require very large amounts of memory, however. The maximum depth of the oct-tree and the maximum number of polygons within a leaf node can be changed to make a trade off between memory usage and processing time.

## Force Generation

After a collision is detected, the forces must be presented. When the cursor touches an object, the specific polygon that was initially touched becomes the active polygon. The forces are in the normal direction and are proportional to the penetration depth into an object, which is measured from the currently active polygon. The direction of the force is interpolated at edges as the cursor moves across an object, and as the current polygon changes, to create a smooth feeling across facets. When the penetration depth of the active polygon becomes negative, the cursor has left the object, the forces are discontinued, and the collision detection algorithm is used again.

### *Normal Direction for Polygons*

An initial issue is encountered because of the nature of a polygonal data set. The outward direction on a polygon is determined from the ordering of the points it contains. The direction that is considered outward is important for both graphics and haptics. In graphics, the outward direction is used to determine shading effects. In haptics, the outward direction is used in collision detection, force direction, and in interpolating between polygons.

A typical way to overcome this problem, which has become a standard in graphics, is to pre-process the points and order them so that all the vertex listings are consistent. If a data set is not already vertex-ordered correctly, then enclosed objects can be checked to make sure the outward direction remains consistent over a surface.

To find out if the vertices in any given polygon are ordered correctly, a point, point 'A', is projected from the center of a polygon in the normal direction of that polygon to a sphere that encloses all of the polygonal objects, to give point 'B' as shown in Figure 3.

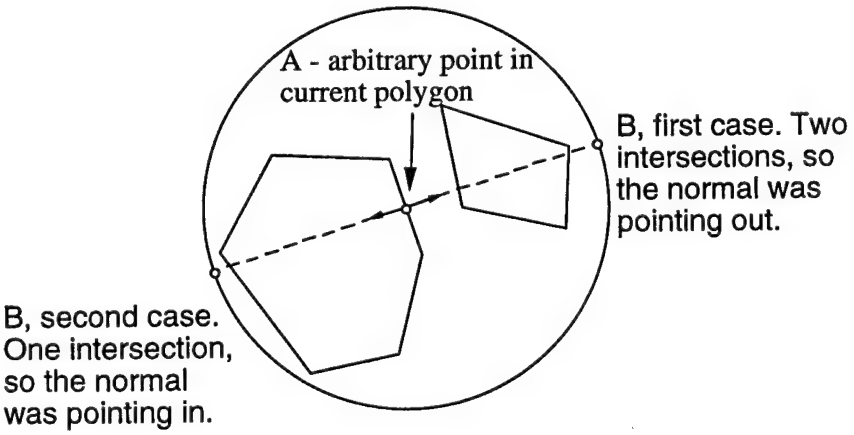



**Figure 3: Finding if a polygon's vertices are correctly ordered. The two cases of point B represent two different ordering conventions.**

Then the number of polygons that are intersected by the segment from point A to point B are counted. When objects are enclosed, an even number of intersections implies that the original normal was pointing outward and an odd number of intersections means that the normal was pointing inward.

### *Determining the Active Polygon and Penetration Depth*

Once a collision is detected, one would then like to be able to slide the cursor from one polygon to another to touch the entire object. Several problems arise while trying to do this. First, the active or

current polygon must always be known so that the direction of the force can be determined. Second, while sliding across polygons, if there is a sudden change in magnitude or direction of the normal force, then corners feel sharp and distinct even when there is only a slight angle between the adjoining polygons.

Originally, the transition from one polygon to another was accomplished by finding the distance from the cursor to the three edges of the polygon's normal projection. If the cursor crossed the projection then there would be a new active polygon. The problem with this approach was that the distance from the cursor to the current polygon's surface would change when changing polygons, and a small jerk would be felt even when the normal direction was interpolated correctly as shown in Figure 4a. This is a problem not found in graphics interpolation because a second variable, depth, is included in the overall interpolation. Also, the distance to the edge of the plane would change, which is used in interpolating the direction of the force. And finally, there can be places within an object that are not in any polygon's projection.

Therefore, a different method was determined in which the distances to 'edge' planes rather than the normal planes were found (Figure 4b). An edge plane, for a given edge, is determined from the two polygon vertices defining the edge and a vector that is the average of the normals of the polygons sharing the edge.

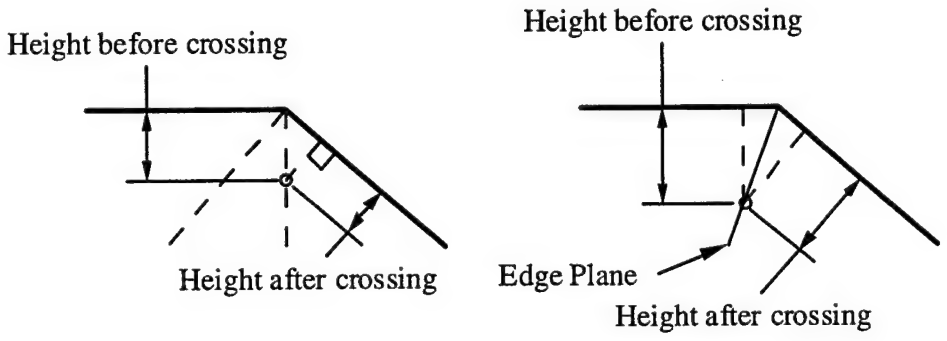
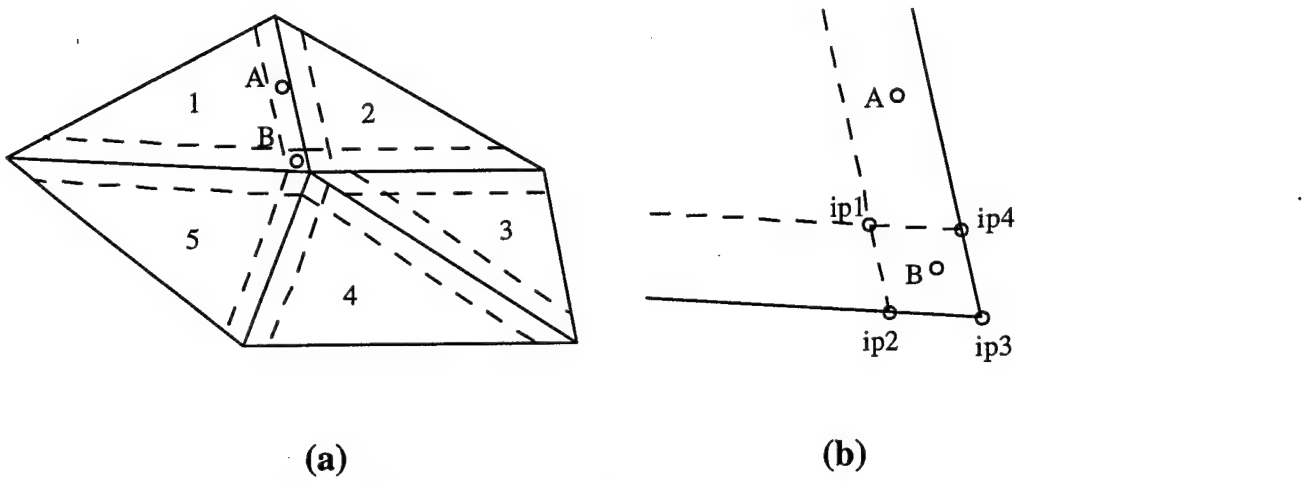



**Figure 4: Determination of a change in the current polygon.**

The method shown in Figure 4b makes the penetration depth and distance to the edge planes consistent while sliding to another polygon. When a change in the active polygon is detected, the new active polygon can be found by finding the other polygon that contains the two vertices in the edge plane that was crossed. This information can be preprocessed and stored in an array rather than finding it as the cycle runs, which saves on cycle time.

***Force Direction Interpolation***

In addition to consistent depth of penetration distances, the directions of the forces need to be consistent to keep the edges smooth. This is accomplished by interpolating between adjoining polygons in a way similar to Phong shading in graphics. When the projection of the cursor into the active polygon comes within a fixed distance from the edge planes, the normal direction is interpolated and then normalized.




**(a)** **(b)**

**Figure 5: Interpolation of the normal direction.**

The direction at point 'A' is interpolated continuously (not necessarily linearly) between polygons 1 and 2. At point 'B', the normal is interpolated from 4 points, ip1 through ip4, all of which are normalized. Ip1 is the normal direction of polygon 1. Ip2 is the average of the normals of polygons 1 and 5. Ip4 is the average of the normals of polygons 1 and 2. Finally, Ip3 is the average of the normals of all five polygons. Care must be taken to make sure the direction is continuous while interpolating over boundaries.

Overall, the interpolation over the edges of the polygons presents an interesting psychophysical effect. If the forces are interpolated correctly, then the shape of the object is perceived more from the direction of the forces than from the cursor's actual position. For example, the facets of a polygonal sphere are easily distinguishable with no interpolation. With the introduction of interpolation, the facets become less noticeable and the sphere feels rounder. As the area over which each facet is interpolated increases, until the forces over the entire facet are interpolated, the sphere appears to 'fill out'. Although a user is still feeling a faceted sphere, it appears completely round and smooth. This can be a powerful effect as the interpolation can be used to modify the way that an object's shape is perceived. Additionally, if an edge is supposed to be distinct, then the interpolation can be turned off which will produce a sharp edge.

***Acute Edges***

If the tool is touching the outside of an acute edge the force will not be interpolated across that edge, which will make the edge feel distinct and sharp, rather than curved. The force is interpolated across the edge in Figure 6a. The force in Figure 6b is taken from the current polygon only. Additionally, there is an issue with acute edges known as the thin-wall problem. This is a common problem in haptic rendering algorithms where the cursor can unintentionally push though a thin wall, such as a knife blade. To handle this problem in the ActivePolygon algorithm, the edge plane that is normally used to transition to a new active polygon, as shown in figure 6a, is not computed or used for acute angles, as shown in Figure 6b.

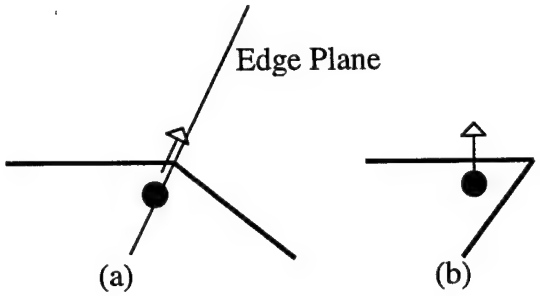



**Figure 6: Interpolation force for an exterior acute edge.**

If the tool is on the inside of a very acute edge it may travel some distance from the polygon before approaching the edge plane, which can create inconsistencies in the forces. Figure 7 shows a cursor point that has touched an interior edge of an acute angle, and has penetrated into the object.

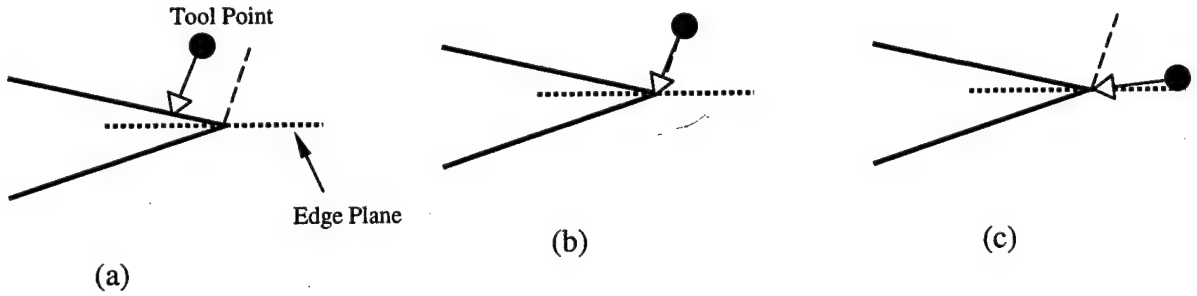



**Figure 7: Interpolation force for an interior acute edge.**

The point is within an orthogonal projection of the polygon in Figure 7a, so the force is in the polygon's normal direction. In Figure 7c, the tool has moved outside the polygon's orthogonal projection, so the force is directly towards the edge. At the point where the direction of the force changes from the normal direction of the polygon to the edge, as shown in figure 7b, both of those computations are equal, thus giving a consistent force. To increase the speed of the algorithm, acute edges are marked during pre-processing.

***Three or More Polygons Sharing a Common Edge***

As has been described, the ActivePolygon algorithm utilizes a local region when computing forces. The currently active polygon and the polygons sharing its vertices are used to create forces based on a single point. In order to speed up the algorithm, and reduce processing time, polygonal neighbors are preprocessed. However, when a single edge is shared by three or more polygons, special processing must be done. In this case, the neighbor information is dependent on the side of the polygon. When three or more polygons share a common edge, one side of a polygon has one neighbor, and the other side of the polygon has another neighbor. If an edge has only two polygons attached to it then the front and rear face neighbors are the same.

The side of the polygon which is currently active will determine which of the two neighbors is used for processing.

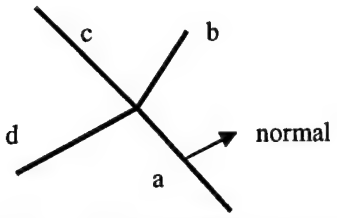



**Figure 8: An edge with three neighbors.**

Shown in Figure 8 is an edge that is shared by 4 polygons. Polygon a will have polygon b as a front neighbor and polygon d set as a back neighbor. Polygon c will not be a neighbor of a.

***Bendable Polygon Algorithm***

As the forces are presented to the user, the visual aspects of the virtual object must be consistent with the object. A compliant object should deform as the cursor moves into it. If the object does not deform, then the cursor can be lost visually within it. One way to solve this problem is to simply project the visual cursor, in the direction of the force, to the surface of the object.

However, if the object is very compliant, then this can create a discrepancy between the visual and haptic senses. To solve this, the Bendable Polygon technique is used. When the cursor first touches a polygon, it is split into 6 different polygons as shown in Figure 9. The cursor is projected normally to the plane of the active polygon, and then that point is projected to the edges of the polygon, making base points that are the framework for the approach. The 'edge base points' move as the cursor moves, always normally projected to the sides of the current polygon. When the polygons are Gouraud or Phong shaded, the edges look smooth and the polygon seems to bend. The effects of the Bendable Polygon technique decrease with increased graphical detail, but for relatively large polygons, the effect works well and allows for lower levels of detail on a deformable object.

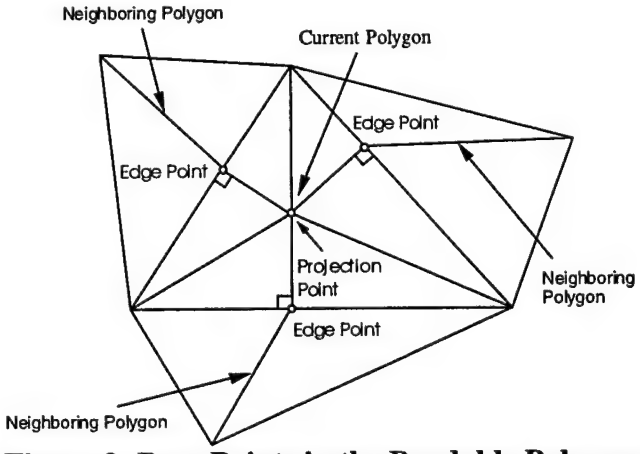



**Figure 9: Base Points in the Bendable Polygon algorithm.**

The cursor is then connected by a spring and a damper to each of the vertices in the active polygon and to each of the edge points. The cursor does not have any forces applied to it by these springs. All six of those points, in turn are connected to the base points shown in Figure 9, also by springs and dampers.

When the cursor moves into a polygon, the vertices (open circles) are pulled away from their respective base points (filled circles), making the polygon bend as shown in Figure 10.

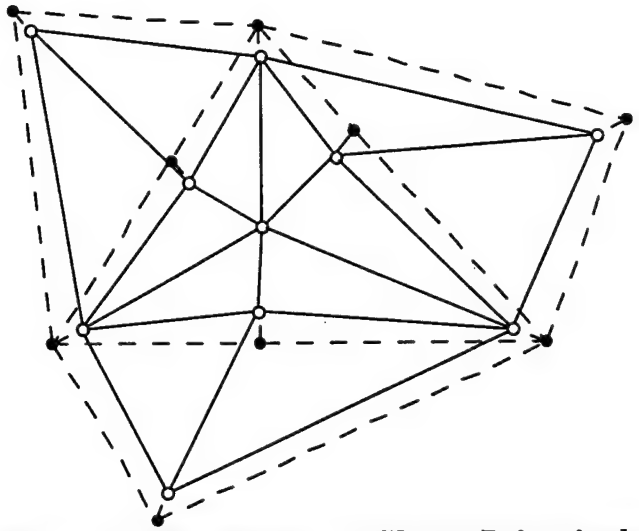


**Figure 10: Base Points and Vertex Points in the Bendable Polygon algorithm.**

The dashed lines represent the object while it is not deformed, and the solid lines represent the polygons that the user sees after it is deformed. As the cursor moves towards an edge, the springs from the edge points to their respective base points lose strength, so the indentation in a polygon remains consistent even as the cursor moves across different polygons. Different spring constants and different levels of strength reduction give different amounts of indentation (i.e. a small indentation on a water balloon as opposed to a larger indentation on a trampoline). The springs from the corner vertices to their base points work similarly, in that they lose strength as the cursor approaches them, so the indentation remains consistent at the corners of the polygons as well.

Then, each of the vertices throughout the object is connected by springs and dampers to each of the vertices touching it, to give an overall ability of large-scale deformation. In large data sets, the number of calculations would become extremely large, so springs might, for example, only be connected to vertices in the general surrounding area of the cursor, depending on the application, or a finite element analysis can determine the object's deformations.

Because the edge points split the current polygon, each of the 3 neighboring polygons must be split into two polygons as well so that there is no gap (Figure 10). The graphics loop therefore draws 6 polygons in place of the original, 2 polygons in place of each neighboring polygon, and then draws all of the rest of the polygons.

The graphical material effects are accomplished by finding the normals for each vertex, an average of all the normals of the polygons containing that vertex. Then the surface is either Phong or Gouraud shaded according to those normals.

***Deformations***

The base points in the polygonal algorithm can be given dynamics properties to allow the deformation of an object. This can be accomplished in several ways. Vertices can be given dynamics properties as described above in the dynamics section, with a very small time variable. In this way, the vertices move slowly and create a feel similar to clay. Additionally, the vertices can be moved with a simpler

method of simply displacing them proportional to the force presented. This however does not allow as much flexibility in modifying the feel of the object.

There are several issues that arise when the vertices are allowed to move. First, there must be some constraints applied to the vertex movements so that a polygon does not collapse on itself. This can be done by maintaining a minimum distance for a side on a polygon, or by allow the vertices to move in only a specified direction along a line.

An additional problem comes from the oct-tree based culling which allows real time collision detection. If vertices are allowed to move then the pre-processed tree structure, that describes which polygon should be checked for collisions given the cursor is in a specific leaf node, can be made invalid.

## Results

The ActivePolygon algorithm was tested on a Dual PIII 800MHZ computer with 512MB of memory, a Wildcat 4210 graphics card, and a Phantom desktop haptic device. The haptics load was obtained using the GHOST 3.1 Haptic Load program. Object 1 has a complex geometry with many areas that are often troublesome. Objects 2 and 3 were the same object and had the same topology, a relatively simple topology, but differed only in their resolution and polygon counts. Object 4 had a large polygon count. There were several aspects of the results that were significant. First, the haptic load averages and peaks were consistent across objects with varying polygon counts. This was expected as the computations are based on a pre-processed data set and are computed locally so the overall size does not affect the local computations. Second, the haptics were stable across varying and complex geometries. Third, the load times increased as the objects had more polygons. This also was expected as there was therefore more data to be preprocessed. A majority of the load time is due to the preprocessing. After the preprocessing is done once, the preprocessed data can be saved with the object, and the load time can be perceptually eliminated. We additionally tested an object with over 1 million polygons. The haptic load peak occurred when initially touching or leaving that object, but maintained a consistent average.

**Table 1: ActivePolygon results**

| Object | Polygons | Load Time (sec.) | Visual FPS | Haptic Load Avg. | Haptic Load Peak |
|---|---|---|---|---|---|
| 1 | 5706 | 1.03 | 21.33 | 20% | 20% |
| 2 | 134232 | 19.54 | 6.76 | 15% | 20% |
| 3 | 239694 | 33.52 | 3.8 | 15% | 20% |
| 4 | 1063452 | 195 | 0.98 | 15% | 80-100% |

**Table 2: Comparison with GHOST polygonal renderer**

| Object | Polygons | Load Time (sec.) | Visual FPS | Haptic Load Avg. | Haptic Load Peak |
|---|---|---|---|---|---|
| 1 | 5706 | 0.9 | 15 | 20% | 30% |
| 2 | 134232 | 11.74 | 6 | 85% | 100% |
| 3 | 239694 | 17.12 | 6 | 70% | 70% |
| 4 | 1063452 | 220 | | Unstable haptics | |

We also compared the ActivePolygon algorithm to the TouchVRML polygonal renderer contained in the GHOST 3.1 API from SensAble Technologies. The listed TouchVRML Visual Frames Per Second are highly qualitative and were obtained by viewing and estimating. In objects 1, 3, and 4 using TouchVRML, the phantom would have force kicks in certain manipulation situations. Objects 3 and 4 had unstable haptics in the TouchVRML application.

Overall, the ActivePolygon algorithm worked very well. Further research is being done to continue to extend its functionality and scope, add modifications such as haptic textures, and integrate it into software applications. The source code for the ActivePolygon algorithm is available in the e-Touch library on the e-Touch web site, www.etouch3d.org.

## Acknowledgements

The writers would like to acknowledge Walt Aviles, Sandia National Laboratories, Arthurine Breckenridge, George Davidson, Tom Caudell, and Tom Furness for their help in the development of this work.

## References

[1] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles, "Haptic Rendering: Programming Touch Interaction with Virtual Objects", ACM Symposium on Interactive 3D Graphics, Monterey, CA, 1995.

[2] T. Massie and K. Salisbury, "The Phantom Haptic Interface: A Device for Probing Virtual Objects", Proceedingsmof the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Chicago, IL, 1994.

[3] SensAble Technologies Inc., "The PHANToM" literature from SensAble Technologies.

[4] C. Zilles, J. Salisbury, "A Constraint- based God-object Method for Haptic Display", IEEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction and Cooperative Robots, 1995.

[5] P. Buttolo, B. Hannaford, B. McNeely, "Introduction to Haptic Simulation", Tutorial 2A, IEEE/VRAIS tutorial notes, March, 1996.

# Expanding the Haptic Experience by Using the PHANToM Device as a Camera Metaphor

Joan De Boeck, Chris Raymaekers, Karin Coninx
Expertise Center for Digital Media
Limburg University Center
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{joan.deboeck, chris.raymaekers, karin.coninx}@luc.ac.be

***Abstract:*** *Many applications that support force feedback make use of a haptic device (such as the PHANToM) used for pointing operations, combined with a second device, mainly used for navigation (such as a 3D mouse). In this research we formally assess the user's performance in a setup in which we use the PHANToM device for camera manipulations. This allows us to eliminate the second device and to free the user of the mental load to drive two different devices. Additionally, when using Sensable's PHANToM as a camera device, we will look into the effect of the additional force feedback.*

## 1. Introduction

Most desktop haptic applications consist of a two-device setup. The haptic device, mostly manipulated with the dominant hand, is used for pointing and manipulation operations. A second device (typically a 3D mouse) is used for navigation operations.
In the context of our research, very little can be found in literature about integrating forces in camera manipulations. In the early 90's efforts have been made in defining the best navigation metaphor [1] for a certain task in a virtual world. *Flying vehicle* and *Scene in hand* are the most commonly known. Other work has been done in improving navigation and wayfinding methods in virtual environments [2]. In some research systems hand-held miniatures [3] or Speed-coupled Flying [4] are presented to facilitate the user's interaction. A usability test by T.G. Anderson [5] provides evidence that additional force-feedback results in better performances when compared to the 2D navigation interface of CosmoPlayer. In this paper, based on Anderson's work, we introduce a fairly new variant on the "Eyeball in hand" navigation metaphor presented in [1] and focus on the comparison with the LogiCad Space Mouse [6].
In the next section we explain our "Camera in Hand" metaphor as a variant of the "Eyeball in hand" navigation metaphor. Afterwards, the experimental setup used to assess users' performance when navigating with this metaphor is described. The results of the formal evaluation are summarized and discussed. Finally, conclusions with regard to the usefulness of the proposed metaphor are formulated.

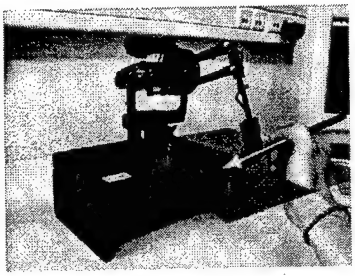


***Fig 1. PHANToM as a camera device with virtual guiding plane***

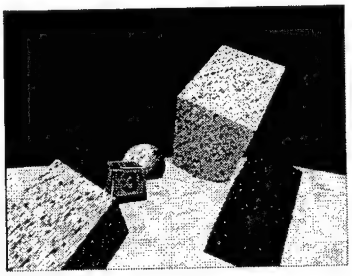


***Fig 2. Virtual arena in which users have to locate the number***

## 2. Eyeball in Hand/Camera in Hand

A possible implementation of the *eyeball in hand* metaphor is to use a Polhemus tracker as a virtual eyeball, which can be moved about the virtual scene. However, this manipulation method appeared to imply a confusing mental model in which disorientation is a common problem. In former research in our lab the *Eyeball In Hand*-metaphor has been extended to a MicroScribe device [7][8]: by moving the MicroScribe's stylus with the non-dominant hand, the virtual camera is repositioned. By defining the viewpoint in such a manner that it matches the direction of the stylus, disorientation will be avoided. As we are not actually handling an eyeball anymore, but rather a pen-like object, we will now call this extension the *Camera In Hand* Metaphor.
This paper adopts the latter metaphor for the PHANToM haptic device, and extends it by applying additional force-feedback. Informal testing taught us to set up a horizontal virtual plane (as shown in fig. 1) as the most useful feedback. This allows the user to easily walk forth and back in this plane. When changing the viewpoint's altitude the user has to act against the resistance of the PHANToM. A formal experiment, described below, was set up in order to formalize and detail the results obtained by informal testing.

## 3. Experimental Setup

The aim of our research was to formally compare this new metaphor and camera device to another existing 3D device. For this comparison condition we have chosen to use the LogiCad SpaceMouse in a "Flying Vehicle" metaphor.
Twenty-two volunteers with mixed experiences in virtual environments participated in the experiment. Most of the subjects are in their late twenties or early thirties, although 4 of them were above the age of 40. All subjects were right-handed and one third of the population was female.
All of the participants had to navigate in a virtual arena to locate and read a digit on a red-white coloured object (see fig. 2). This test had to be performed in three conditions; each condition consisted of 15 trials. The first condition measured performance with the SpaceMouse, the second looked at the PHANToM device without force feedback and finally the PHANToM device with force feedback has been tested. To eliminate transfer effects, the order in which to take the experiments was counterbalanced. During each trial the elapsed time and the total traveling distance had been logged. Finally, at the end of the test a comparative questionnaire had to be filled-up by the subjects. We have to note that one of our 22 test persons had trouble in performing the tasks in all conditions. Since his results exceeded 12 times the standard deviation, we have omitted those values.

## 4. Results

Chart 1 shows us the median values of the completion times of all subjects, per trial in each condition, which gives us a first impression of the results.

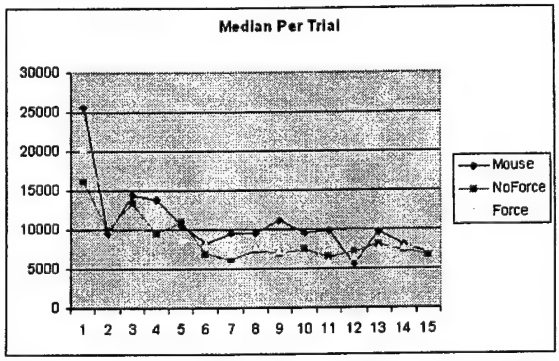



***Chart 1. Completion Time (ms). Median values per trial***

In our further analysis[1], we consider the first 5 trials for adaptation to the proposed input devices, and so leave them out of our computations[2]. As can be seen from chart 1 and table 1, the average completion time in both PHANToM conditions is slightly better than the SpaceMouse. With a P-value of 0.12, there is no significant difference, however.

| | |
|---|---|
| Mouse | 13333 ms |
| PHANToM Force | 10256 ms |
| PHANToM NoForce | 9499 ms |

| **P-Values** | |
|---|---|
| Condition[Mouse-PH no] | 0.1231 |
| Condition[PH Fo - PH no] | 0.8241 |

***Table 1. Averages and P-values over all subjects***

Because of the relative heterogeneity of our population, we have divided all subjects in four categories depending on their experience in 3D navigation: no, little, much and very much experience. Statistically, the groups with little, much and very much experience behave the same. Therefore, in our further analysis, we consider two levels of experience: novice (users without any 3D navigation experience) and experienced (all the others).
If we look at the average completion times in table 2, we can see there's still no significant difference between any of the conditions in the experienced group. However, we now notice a strong significant difference in completion times between the SpaceMouse and the PHANToM conditions among the novice users.

| | **Novice Users** | **Experienced Users** |
|---|---|---|
| Mouse | 17263 ms | 9760 ms |
| PHANToM Force | 9381 ms | 11060 ms |
| PHANToM NoForce | 9007 ms | 9941 ms |

| **P-Value** | | |
|---|---|---|
| Condition[Mouse-PH no] | <.0001 | 0.4922 |
| Condition[PH Fo - PH no] | 0.0507 | 0.2636 |

***Table 2. Averages and P-values per category***

A subjective questionnaire, filled up by all subjects after the test, shows us that experienced users significantly prefer the SpaceMouse over the PHANToM. On the other hand novice users choose one of both PHANToM conditions.

| | Novice | Expert |
|---|---|---|
| Mouse | 2 | 10 |
| NoForce | 5 | 1 |
| Force | 3 | 0 |

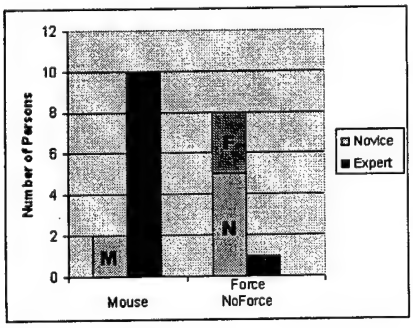



***Fig. 3. Subjective preference per category***

## 5. Discussion

As can be seen from table 2 experienced users objectively do not perform different in one or the other condition. If we look at the measurements of the novice users, we see a dramatic improvement when using the PHANToM. Compared to the values of the experienced category, we can notice that the values of the novice users using the

[1] Using ANOVA
[2] This is supported by our results, as can be seen in chart 1.

PHANToM condition are similar. This means we can conclude that our *Camera In Hand* metaphor provides a possibility for the inexperienced user to perform equally to their experienced colleagues.
However we can also conclude that the addition of force-feedback, which implements a horizontal guiding plane, doesn't offer any advantages. There's even advantage for the no-force condition, though the difference is insignificant.
As the performances of the experienced users are similar in all conditions, we have to ask why they collectively choose for the SpaceMouse condition. Our experienced users all spend several hours a day on a computer and all have their 3D experience playing games with mouse and keyboard. For that reason we suspect those users to have certain expectations and so feel more familiar with the SpaceMouse. In addition, some of those users report the limited workspace and tiring pose when using the PHANToM as a disadvantage.

## 6. Conclusion and future work

In this work we presented a 3D camera metaphor using the PHANToM device with and without force feedback. This metaphor can eliminate the use of a second input device in a haptic setup. The performances of those conditions have been measured and compared to the navigation with a SpaceMouse in a formal usability test. As a conclusion we can state that, using the PHANToM, novice users act in the same way with the *camera in hand* metaphor as the experienced users. When those users are using the SpaceMouse there is a strong performance penalty. However, experienced users mostly choose for the SpaceMouse, while they perform equally in all conditions. Finally, we also can conclude that additional force feedback in a sense of an additional guiding plane doesn't offer any benefits in this test.
We believe the *camera in hand* metaphor will be of interest to introduce the novice user into 3D environments, but it can also be useful when manipulating a scene that is rather centralized in a limited volume. Although additional force feedback doesn't seem to offer any benefits, we think, dependent on the task, additional stability can be obtained by finding a appropriate force factor, which is possibly somewhat smaller than the resistance force in our test.
In our future work, we want to evaluate the effect of additional functionality to step out of the limiting workspace of the PHANToM. This can be achieved by e.g. homing the PHANToM without changing the virtual camera, or by moving the virtual camera when the users pushes the outer limits of the PHATNoM's workspace [5].

## 7. Acknowledgements

Part of the work presented in this paper has been subsidized by the Flemish Government and EFRO (European Fund for Regional Development).
We also want to thank the people of CIT Engineering and Porta Capena and all other volunteers for their participation in this test.

## 8. References

[1] C. Ware, S. Osborne. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. Computer Graphics 1990 Vol 24 Nr 2
[2] G.A. Satalich. Navigation and Wayfinding in Virutal Reality: Finding Proper Tools and Cues to Enhance Navigation Awereness. http://www.hitl.washington.edu/publications/satalich/home.html
[3] R. Pausch, T. Burnette. Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures. Computer Graphics 1995, Annual Conference Series, p 399-400
[4] D.S. Tan, G.G.Robinson, M. Czerwinski. Exploring 3D Navigation: Combining Speed-coupled Flying with Orbiting. CHI 2001 Conference Proceedings p. 418-425, Seattle, Washington, USA
[5] T.G. Anderson. FLIGHT: An Advanced Human-Computer Interface and Application Development Environment. Thesis Master Of Science, 1998, University of Washington.
[6] LogiCAD SpaceMouse; http://www.logicad3d.com/products/Classic.htm
[7] T. De Weyer, K. Coninx, F.. Van Reeth Intuitive Modelling and Integration of Imaginative 3D Scenes in the Theatre, Proceedings VRIC 2001 p 167-173
[8] Immersion Microscribe 3D: http://www.immersion.com/products/3d/capture/overview.shtml

# Scene Complexity: A measure for real-time stable haptic applications

Eric Acosta, Bharti Temkin
Department of Computer Science, Texas Tech University
Bharti.Temkin@coe.ttu.edu

**Abstract**

We discuss real-time issues of scene-complexity in order to frame a technical envelope for stable haptic applications. How large a scene can a haptic application support? Specifically, we try to determine the largest stable haptic scene possible when GHOST is used to develop an application. The scene consists of a number of non-overlapping primitives or polymesh objects. The scene complexity is measured as a number of primitive objects or polygons in polymesh objects that allow stable haptic interactions.

The initial data collected indicates that earlier versions of GHOST allow for more complex scenes. Each later version seems to reduce the number of objects that can be included in a scene. For example, Version 1.2 allows for inclusion of 1100 non-overlapping spheres in a stable haptic application. Version 2.0 allows only 770 (almost 30% less) such objects, while version 3.0 allows only 600 (almost 45% less). The system used for data collection is a Pentium III 500 MHz computer with 256 MB of RAM. This machine has an nVidia Riva TNT2 Ultra video card with 32 MB of memory and runs Windows NT Workstation 4.0 with Service Pack 6.0.

In order to quantify and understand these observations, we have developed an application that estimates the distribution of resources used within the haptic loop; the fractions used by collision detection, graphics, and haptic processes. This study provides a method for performance analysis of haptic systems. Scene complexity plays a key role in the creation of haptic applications by providing critical data needed to estimate the feasibility and performance limits for generic haptic environments. It thus should have a broad impact on the design of haptic applications.

## 1. Introduction

Scene complexity is a measurement on how complex a scene can get and still allow for stable haptic interactions. The real-time performance is typically reduced in proportion to the increase in the complexity of the scene. In fact, it is well known that the time to compute haptic interactions increases with the number of polygons. The approach taken in previous work was to make the haptic servo loop rate essentially independent of the number of polygons [1]. However, this invariance is obtained only after the contact is made with an object and while the object is being touched in the neighborhood of the proxy, while the proxy remains inside the object, making this an efficient haptic rendering technique for a single polymesh object. In this paper, the scene complexity is measured as a number of primitive objects or polygons in polymesh objects. By understanding the limits imposed by the complexity of the scene, we can estimate the feasibility and performance limitations of generic haptic environments.

First, we consider the greatest lower bound (GLB), the highest scene complexity for which the application always runs with stability and no errors. Next we consider the least upper bound (LUB), where some instability and errors are allowed to occur. When the number of objects is between GLB and LUB the application runs some of the times and produces errors or instabilities at other times. When the complexity exceeds LUB of the scene, the application produces errors instantly after haptics is initialized. This establishes a scene complexity stability range for haptic applications, as shown in Figure 1.

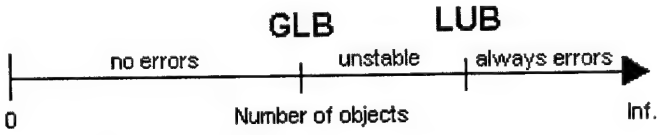



Figure 1: Scene complexity stability range.

## 2. Primitive Objects: Spheres and Box complexity tests

To establish the scene complexity, our application allows a tester to specify the number of primitive objects (in the x, y, and z directions) in the scene. The application automatically creates and spaces the objects to prevent them from overlapping as shown in Figure 2. The use of non-overlapping objects eliminates other factors affecting haptic load (hload) [2]. Hload is the time required to complete a haptic loop. Our application records the hload data with $10^{-3}$ ms precision and stores it in a file. In order to estimate the time distribution of the graphics, haptic, and collision detection tasks [3] within the 1 ms haptic loop, hload can be measured in several modes: touching (T) or not touching (NT) an object, graphics on (G) or off (NG), and removing geometry from the haptic scene graph (NH). By removing the geometry branch "hapticScene", as seen in Figure 3, the geometry can be eliminated from the haptic scene graph. The NH mode represents the time required to perform other duties of the haptic loop, such as the device position query and the scene graph traversal, without having to perform any collision detection on the geometry objects themselves.

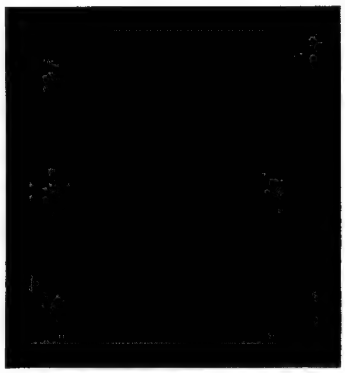


Figure 2: Spheres in a 3x3x3 (x,y,z) fashion.

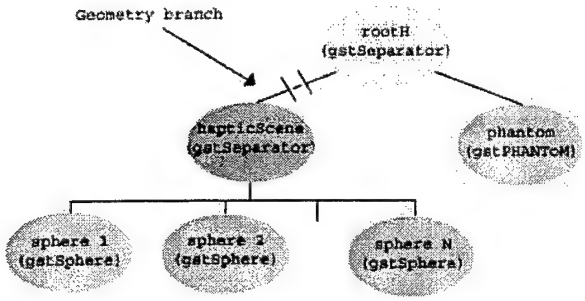



Figure 3: Sample scene graph representation.

The test data was collected five times for a combination of modes between G or NG, T or NT, and NH (G_T, G_NT, NG_T, NG_NT, G_NH, NG_NH). Starting with 8 (2x2x2) objects, the number of objects is increased by one in each direction (e.g. next test has 9(3x3x3) objects), until an error is generated for exceeding the 1 ms time constraint of the haptic duty cycle. At this stage, the number of objects is slowly decreased to identify the GLB. Next, the number of objects is increased to find the LUB, a point at which the application instantly produces an error.

Tests were conducted using three different versions of GHOST (1.2, 2.0, and 3.0) to evaluate performance changes from version to version. The GLBs and LUBs for the different versions are summarized in Table 1. The GLB between version 1.2 and 2.0 is reduced by about 30% and reduced by another 22% between versions 2.0 and 3.0 for spheres. Test results for version 3.0 are displayed in Figure 4. From the tests, we calculated that on average displaying graphics increased the hload by about 2-5% and touching an object increased it by about 7-18%. Our test results indicate that earlier versions of GHOST allowed for more complex scenes. Each subsequent version reduced the number of objects that could be included in the scene for a stable haptic application. The GLB between versions 1.2 and 2.0 is reduced by about 27% and is further reduced by 25% between versions 2.0 and 3.0 for boxes. In general our test show that hload increases

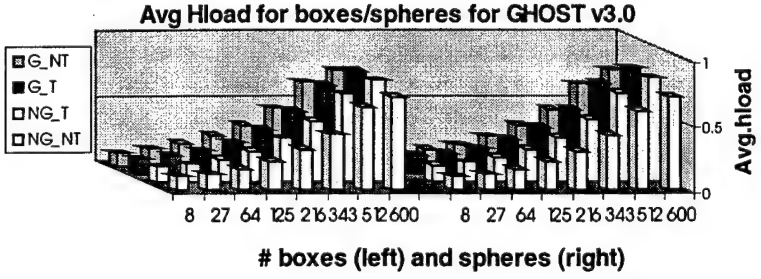



Figure 4: Avg Hload for GHOST v3.0

| Version | Object | GLB | LUB |
|---|---|---|---|
| 1.2 | Box | 1100 | 1188 |
| | Sphere | 1100 | 1200 |
| 2.0 | Box | 800 | 847 |
| | Sphere | 770 | 847 |
| 3.0 | Box | 600 | 729 |
| | Sphere | 600 | 729 |

Table 1: GLB and LUB for GHOST versions

fairly linearly as objects are added to the scene graph for every version of GHOST. Clearly, overhead was introduced from version to version. In case of no graphics and no haptic scene (NGNH), the average hloads were .05ms, .08ms, and .09ms for versions 1.2, 2.0, and 3.0 respectively. The data represents the time required to perform basic duties of the haptic loop, such as device position query and scene graph traversal for nodes other than geometry.

The hload percentage for collision detection is about 82-83% for spheres and 89-90% for boxes, for graphics it is about 3% for both spheres and boxes, and for touching an object it is about 11-13% for spheres and 6-8% for boxes.

## 3. Polymesh objects complexity tests

For a single polymesh object, a box made up of length, width, and height segments was used. These segments define the resolution of the object and determine the number of polygons that form it. The number of vertices (*nv*) and faces (*nf*) can be calculated as follows, where *l*, *w*, and *h* are the number of length, width, and height segments respectively:

$$nv = 2\times [(l+1)(h+1) + (w-1)(h+1) + (w-1)(l-1)]$$

$$nf = 4\times[(l\times h) + (w\times h) + (w\times l)]$$

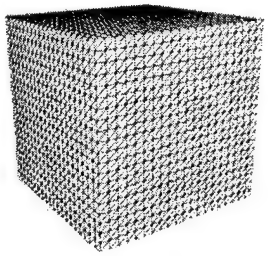


Figure 5: Polymesh box

Figure 5 is a wire frame rendering of the box showing how the object is made of triangular polygons. Starting with a 10x10x10 segment (602 vertices, 1200 polygons) box, the numbers of segments (length, width, and height) are increased by 10 for each test.

Figure 6 displays the average results of five test runs with GHOST version 3.0. According to these results, the hload varied by very little when the polygon count increased for a single object. However, when polygon counts exceeded about 120k, GHOST started to produce random errors, even though hload was under 0.2ms. From these results, for a single object with no overlapping polygons, we found the GLB and LUB to be 120k and 235k polygons respectively.

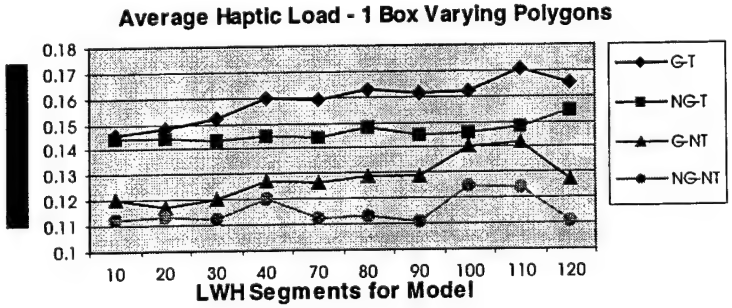



Figure 6: Avg test results for increasing polygons for one object

For multiple polymesh objects, the number of polygons is kept consistent with each increment step in the previous test. Multiple 10x10x10 segment boxes, each containing 1,200 polygons, are used to form the polygon count for each test run. The vertex count was greater in the multiple polymesh objects test than in the single polymesh object test by 2·(N-1) vertices, where N is the number of boxes. The increase in hload is relatively linear as objects are added to the scene, Figure 7. Further testing gave the GLB and LUB to be 35 objects (42k polygons) and 49 objects (58.8k polygons) for these polymesh objects with non-overlapping polygons. The limit of 1ms was exceeded for 49 objects, therefore not allowing for the true value to be recorded. However, we did notice that with only 35 objects, the

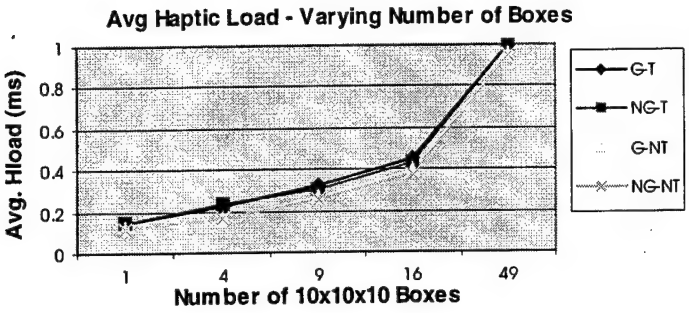



Figure 7: Avg test results for increasing number of objects

number of polygons from the single polymesh object test was reduced from 120k to 42k in the multiple polymesh objects test. Even with an additional 78k polygons the maximum hload was about 80% less.

We estimate that on the average, collision detection for the polymesh objects is about 71-77%, while graphics is about 2-8%, and touching the object is about 19% of the hload. Tests also showed that the graphics increased hload dependent of the number of polygons. Touching the polmesh, however, raised hload on average by about 26%.

## 4. Conclusion

In this paper, we addressed scene complexity issues based on the maximum number of primitive objects as well as the maximum number of polygons for single and multiple polymesh objects that allow stable haptic interactions when using GHOST. We discussed the idea of specifying a scene complexity stability range. This range was defined by the greatest lower bound (GLB) and the least upper bound (LUB). GLB is the highest scene complexity for which the application always runs with stability and produces no errors. The LUB is a point at which the haptic application will no longer run and produces errors instantly after haptics is initialized. When the scene complexity is between GLB and LUB the application runs, but is unstable and prone to errors. We were also able to estimate the percentage of hload used for the collision detection, graphics, and user touching an object.

Though these initial tests gave us complexity bounds for some haptic applications, more tests need to be performed in order to estimate the feasibility and performance expectations for a general haptic environment that involves many other levels of complexity. For example, other tests results indicate that hload increases if the point of contact is within multiple bounding boxes and increases even more if the point is touching multiple objects. Tests also show that there can be different hloads within a single polymesh object, depending on its topology (eg. corner points, overlapping polygons, etc.). Understanding scene complexity limits plays a key role in creation of haptic applications by providing critical data needed to estimate the feasibility and performance for generic haptic environments.

## 5. References

[1] Ho, C., Basdogan, C., Srinivasan, M.A., 1999, "An Efficient Haptic Rendering Technique for Displaying 3D Polyhedral Objects and Their Surface Details in Virtual Environments", October 1999 Vol. 8, No. 5, pp. 477-491, Presence: Teleoperators and Virtual Environments.

[2] Acosta, Eric J., Haptic Virtual Environment, M.S. Thesis, Computer Science Department, Texas Tech University, May 2001.

[3] Farida Vahora, Bharti Temkin, Thomas M. Krummel, Paul J. Gorman, "Development of Real -Time Virtual Reality Haptic Application: Real-Time Issues", *12th IEEE Symposium on Computer-Based Medical Systems* - CBMS 1999, June 18-20, pages 290-295, ISBN 0–7695–0234–2

[4] GHOST Software Developers Toolkit Programmers Guide, SensAble Technologies, Inc.

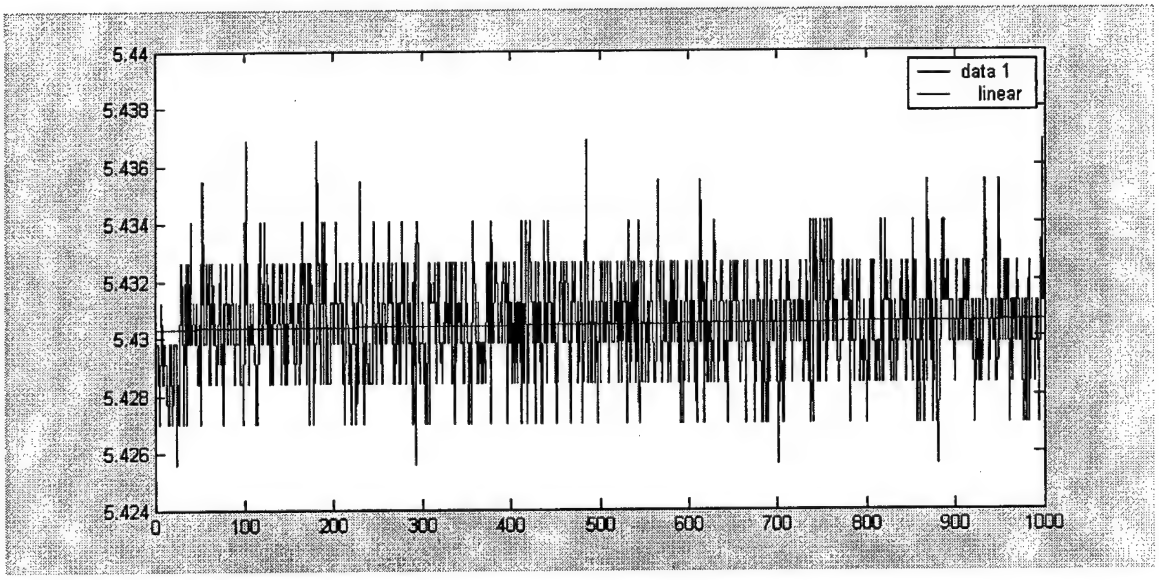
# Dealing with Desktop Gimbal Noise

Karl Reinig  Chris Lee

## *Introduction*

The gimbals of the desktop PHANToM introduce considerable noise into the measured state of the stylus. When the virtual tool tip is represented by a point located at the origin of the gimbal, the noise goes unnoticed. However, when the virtual tool tip is projected away from the origin of the gimbal or the tool possesses non-trivial extent such as a ray or cylinder, the noise noticeably degrades both the haptic and graphic display of the tool. If the tool is being used as a virtual camera, the noise will introduce jitter into the graphic display of the virtual environment. This paper discusses some of the problems and work-arounds, when attempting to filter the noise.

The following figure shows one of the gimbal angles, as reported by the GHOST method getGimbalAngles.




Throughout the one thousand samples, the stylus was resting on a solid surface. The resolution of the encoder, approximately .0015 radians or .086 degrees, can be seen in the discrete steps of the samples. The graph also shows the noise to be about eight times the resolution.

Consider the consequences of projecting the virtual tool ten centimeters from the gimbal origin. Ten centimeters times the tangent of .0015 (the gimbal resolution) results in a positional resolution of .015 cm. But the noise results in a positional jitter of approximately 1.2 mm. This is unacceptable for most applications.

## *Filtering the Noise*

Creating an optimal filter is not the point of this paper. Instead, we present some of the problems we encountered while implementing a simple filter and discuss our current work-around. Consider implementing a filter that simply averages the signal by adding equally weighted samples and dividing by the number of filter samples. The resulting low-pass filter will scale the excursions due to the noise by the inverse of the number of filter samples. It will, of course, also degrade the high frequency fidelity of the virtual tool.

Let $\alpha_i$ be the sample taken $i$ steps before (for example, the current sample would be $\alpha_0$).
Let $F(\alpha)$ be the result of filtering the $\alpha_i$.
For our simple filter,

$$F(\alpha) = \frac{1}{n}\sum_{0}^{n-1}\alpha_i$$

In GHOST, the gimbal state is available as the upper 3 x 3 elements of a 4 x 4 transform. The most straightforward implementation of the filter would perform the average on each of the elements of the 3 x 3 array. However, this is more than just wasteful since the resulting transform would no longer be a rotation matrix. At the very least, you would have to renormalize its elements. This is similar to the problem of using matrices to represent camera motion.

Ideally you would filter the gimbal noise directly and then use it to form the PHANToM transform. Unfortunately, while GHOST does give direct access to the gimbal angles, it does not give access to the methods that create the PHANToM transform. Since the rotational components of the PHANToM transform contain positional information, this is not a trivial transform.

It would be helpful if the GHOST team made a method available to combine gimbal angles and position information to create the transform. This is not necessarily straightforward since some of the required information probably only exists at the driver level. What follows is a less than ideal solution for the interim.

GHOST does make a set of Euler angles available from the PHANToM transform. The angles are the rotations about the x, y, and z axes that would put the stylus in its current orientation. The Euler angles can be used to recreate the PHANToM transform and are therefore candidates for filtering. Unfortunately, the transform that gets the Euler angles from the PHANToM transform does not produce a continuous set of Euler angles. There are places in which each of the Euler angles jump by 2PI. Since a jump of 2PI about any axis has no effect on the orientation, the ambiguity generally goes unnoticed. However, the result of filtering an angle that has 2PI jumps is an angle that makes a quick trip around the quadrants. This causes completely unacceptable rapid movements in the virtual tool. The method described here recognizes the occurrence of a 2PI jump and removes it from the current sample. Note that the gimbal angles themselves introduce no such ambiguity.

## Implementation

A brute force implementation of the averaging filter could be accomplished by keeping an array of samples. Samples would be added to the array using a cyclic index that runs from 0 to n-1. At each time step, the filtered value could be found by summing the elements of the array and dividing by its length. If the averaging is to be performed over 100 steps, each time step would contain the summing of 100 elements.

The summation can be eliminated by developing a recursive implementation. It can be shown that the difference between the filtered samples at consecutive times steps is just one over the length of the filter times the difference between the current sample and the oldest sample. Therefore instead of summing the elements, we need only find the difference between the current sample and the oldest sample, scale it by the inverse of the number of steps, and add it to the previous filtered value.

To eliminate the problems caused by the 2PI jumps, compare the current sample with the previous sample. If the current sample is more than PI larger than the previous sample, simply subtract 2PI from the current sample. If the current sample is more than -PI smaller than the previous sample, simply add 2PI to the current sample. As long as the magnitude of the actual change in the angle is less than PI, there should be no ambiguity.

The PHANToM transform can be constructed from the Euler angles using the following matrix, where X, Y, and Z are the first, second, and third elements of the Euler vector.

$$\begin{pmatrix} \cos Y * \cos Z & \cos Y * \cos Z & -\sin Y & 0.0 \\ \sin X * \sin Y * \cos Z - \cos X * \sin Z & \sin X * \sin Y * \sin Z + \cos X * \cos Z & \sin X * \cos Y & 0.0 \\ \cos X * \cos Z * \sin Y + \sin X * \sin Z & \cos X * \sin Y * \sin Z - \sin X * \cos Z & \cos X * \cos Y & 0.0 \\ posX & posY & posZ & 1.0 \end{pmatrix}$$

## Summary

The gimbals of the PHANToM desktop introduce unacceptable noise into the state of the stylus. The information required to transform the gimbal angles into a useful PHANToM transform are not made available. A solution is to decompose the PHANToM transform into Euler angles, filter them (watching for 2PI jumps), and then reconstruct the PHANToM transform.

# A Dynamic Design Strategy for Visual and Haptic Development

Arthurine Breckenridge
Interaction Laboratory[1]
*Sandia National Laboratories*
arbreck@sandia.gov

Ben Hamlet
*Sandia National Laboratories*
brhamle@sandia.gov

Derek Mehlhorn
*University of Washington*
dtmehlh@sandia.gov

Kevin Oishi
*Carnegie Mellon University*
ktoishi@sandia.gov

## Abstract

*The evolution of modern computer programming languages comes with the need for the strategies with which we implement them to change as well. Fully dynamic and reusable visual and haptic simulations are now possible given the modular nature of current programming languages. The new standards for simulations are dynamic applications that can load and utilize code modules without shutting down, and that will almost never require a complete rebuild for new applications. Two major issues we addressed and implemented in pursuit of this standard are:*

*I. Managing the resources of one machine*

*II. Dynamic Human Computer Interface (HCI)*

*This paper discusses the limitations of current development strategies, and presents the work done at Sandia National Laboratories to develop an application that will conform to the new dynamic standards.*

## 1. Limitations of Current Simulation Strategies

Current Virtual Reality (VR) simulation strategies have two major limitations. First, they do not exceed the resources of one machine. The current strategy for VR simulations of all kinds is to create a specifically tailored application, from the ground up, to accomplish a specific goal. Not only are these applications designed to simulate only one situation or event, but, once completed, they cannot be updated or modified without being completely recompiled. There are two main drawbacks to this development strategy. The applications, or parts of them, are difficult to reuse in future work, and the future simulations must be almost completely re-written. A number of attempts have been made to reduce the need for complete program rewrites. These come in the form of Application Programming Interfaces (API's).

Second, current VR strategies do not support dynamic Human Computer Interaction (HCI). HCI has traditionally been limited to 2-dimensional devices like mice or keyboards. When working with a 3-Dimensional simulation or programming environment, working with a 2-dimensional input device is counter intuitive and limiting to the user. Development in the field of computer haptics has improved HCI in a number of ways, including a much more intuitive virtual environment. Computer haptics has great potential in the VR field, not only for simulation purposes but for simulation development [1] as well. Such devices as SensAble Technologies' Desktop PHANToM[2] [2] provide a user with 6

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

[2] PHANToM is a registered trademark of SensAble Technologies, Inc.

degrees of freedom of motion (x, y, z, yaw, pitch, and roll) as well as 3 degrees of force feedback (x, y, z). Not only do haptics devices, such as the PHANToM, allow users to interact with a computer in 3-dimensions, but they also allow the computer to interact with the user physically, in addition to the standard visual feedback. These devices, being developed into new APIs, have proven to be intuitive and efficient in the development and interaction with virtual environments [1,3]. The improvement of HCI and the emergence of API's have increased programming productivity and reusability, proving a more efficient and desirable design strategy is becoming possible.

## 2. A Dynamic Standard

In the past, simulation development has been limited by the linear nature of early programming languages, such as Fortran and Cobalt. Newer languages, such as C++ and Java, are now object oriented, encouraging modular code development, and support for greater code reusability. The modular nature of current programming languages lends itself well to a new strategy for VR and application design.

One of the limitations, needing to be addressed, of current VR strategies is the lack of reusability. Extremely powerful and impressive applications and simulations have been developed as single use applications. Therefore, these program's contributions to their fields are static. They cannot be easily modified to perform different tasks, nor can they be easily disassembled and applied directly to another application. Current computer hardware and software technology make possible a number of improvements to VR development strategies, including the ability to manage the resources of a machine through dynamic loading.

Simulations can now be designed in a modular nature that allows for dynamic scalability and reusability. The new standard for VR development should be multiple use applications that can dynamically load new features without recompilation. Current API's should be designed in this manner to increase productivity and reusability. The primary benefit of a new dynamic standard is the reusability and expandability that will allow for more robust and extensive simulations than ever before. Dynamically loadable modules can also enable the simulations to be run on a much larger range of machines, since users can dynamically customize the resolution, or complexity, of the application before or during execution. People using the simulation will only load the modules they need or can support with their machine. This strategy will also enable a large number of people to develop VR simulations without being programming or design experts.

The second improvement to current VR development strategies that we worked with lies in HCI. One limitation of current HCI is that, for the most part, it is limited to a single user with a static user interface. Currently, the means by which users interact with their virtual environments do not change with data needs. Therefore, in order to maximize the breadth of an application, all conceivable features are loaded upon execution, a method that will quickly exceed system resources. Our implementation of a dynamic HCI includes the ability to dynamically change the way that one can interact with one's simulation environment by using a series of dynamic menus that can be developed during runtime. The ability for multiple users to work collaboratively over a network in a Virtual Collaborative Environment (VCE) would also be extremely valuable, and has limitless potential. Productivity and quality of products will be increased as designers and customers collaboratively design in a VCE, even if they are hundreds of miles apart.

## 3. An Implementation Example

### 3.1 Overview

An initial attempt to implement the new standards of VR development, as stated above, has commenced at Sandia National Laboratories' Interaction Lab. The basic methods and strategies used are applicable to more general cases of VR development. It should be noted that the machines used in development were little more than commercial-off-the-shelf PC systems, readily available to most individuals interested in the new wave of VR development. Novint Technologies' e-Touch[3] was used as a basis for haptics development with Sensable Technologies' Desktop PHANToM device. Novint is an internet spin-off company, from the Sandia National Laboratories' Interaction Lab. e-Touch is an open module effort to provide a 3D application programming interface as well as a graphic/haptic user interface (GHUI) to the internet community.

[3] e-Touch is a registered trademark of Novint Technologies, Inc. http://www.etouch3d.org

The project is tailored towards an architectural design application. However, since the project is meant to implement the new standards for haptics and visual simulations, the application area goes far beyond a simple CAD program. At the core of the project is the need to manage the resources of a machine, to dynamically add features, ranging from rendered objects to new modules of code, without having to exit the application. Other goals include improving our HCI by supporting multiple users, working collaboratively across a network, using dynamically changing tools and interfaces.

exist within the foundation of the application. New objects, tools, and commands can then be derived from these classes, providing expandability and compatibility. This structure also enables novice users to implement basic features and provides advanced users with the ability to overload the base implementation and create unique and complex additions to the simulation. Objects and tools are also designed in a way that they depend only on the lowest level of our API, Novint's e-Touch. Command modules are used to interface them with specific applications. Therefore, the same objects and tools can be reused in an infinite array of e-Touch

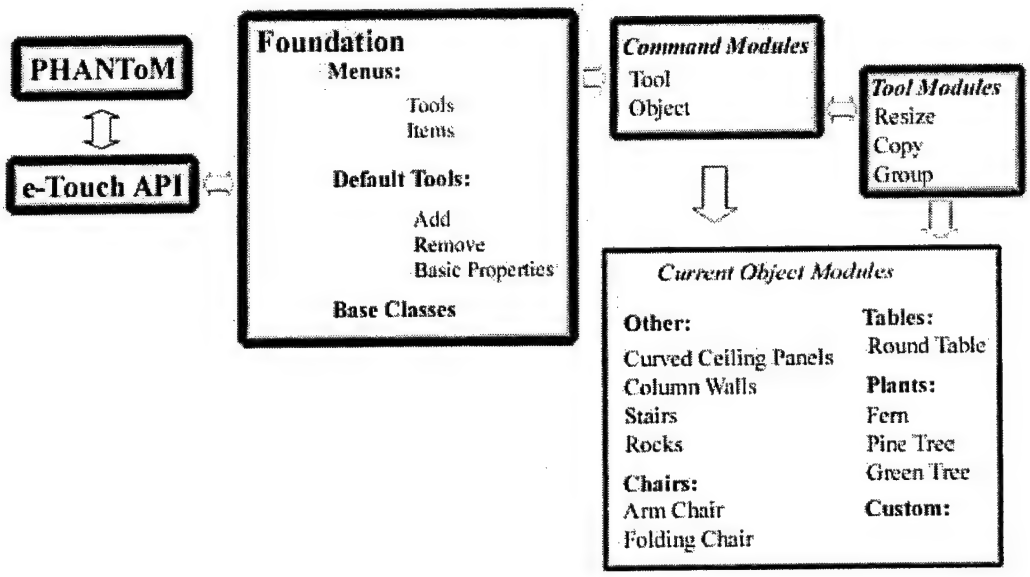



***Chart 1.* Shows the design structure of the application.**

### 3.2 Basic Structure

The application structure breaks down into three major categories: objects, tools, and commands. Objects are graphic and/or haptic items that can be dynamically added, removed, and manipulated within the simulation. Tools enable a user to interact with one's environment and with the objects therein. The tools, except for a select few default tools, and all objects are designed to be independent code modules that can be dynamically loaded upon request. Commands are code modules that are used to interface objects and tools with the foundation of the application.

To support expandability and dynamic loading of different features, a series of base or template classes

based applications by simply swapping out their command modules.

*Chart 1* shows the basic structure of our application. The foundation classes consist of, essentially, the groundwork for objects, menus, and tools. An object is defined as anything rendered with either graphics, haptics, or both. For our purposes in the architectural application, objects are created to use both styles of rendering. Tools are usually represented by a 3D cursor, which is directly controlled by the PHANToM for input. Tools are used to alter the state of the application by adding, copying, deleting, and in other ways manipulating an object. Tools generally manipulate or change some property of an object. Menus are completely three dimensional, and contain 3D objects, such as buttons,

sliders, and knobs. An explanation of the menuing system can be found in section *3.4 Menus.*

At this point, it should be noted that this project's system of combining graphical and haptic representations of objects did not require the redevelopment of 3D modeling applications. The aim is not to replace existing, proven, and refined applications, but rather to address a new method of efficient development that can use and improve upon existing programs. The graphical nature of objects is still designed using more specific, existing, applications. The project currently uses imported 3D Studio Max[4] files to create the graphics for our haptic objects, and contains a framework for converting other types of 3D object files into data that can be used by the application. Using these pre-designed graphics, a haptic representation is either specifically defined or generally applied based on a series of parameters, yielding a completed haptic object.

An important concern with any system that combines haptics and graphics data is the need for an on disk storage system that also ties these two representations together. This can be accomplished through the use of a common file format that stores both object depictions in a single file. Such a format would allow even the most novice users to create simple objects, while more advanced and knowledgeable users would still be able to create the complex, individualized, and ornate objects they require.

### 3.3 Tools

Our application provides a dynamic and 3D GHUI. A series of default tools allow users to interact with their environment instantly in several basic ways, while a series of 3D menus dynamically reconfigure themselves to incorporate new tools during runtime. *Image 1* shows an object being resized using one of the applications default tools. *Image 2* shows the default property manipulation tool. Without any additions to the applications, default tools enable users to add, delete, resize, and manipulate object properties, while still possessing the ability for custom tools to be loaded in at anytime for customized interaction. This system increases productivity, decreases lost time, and allows for a more robust and overall user-friendly application.

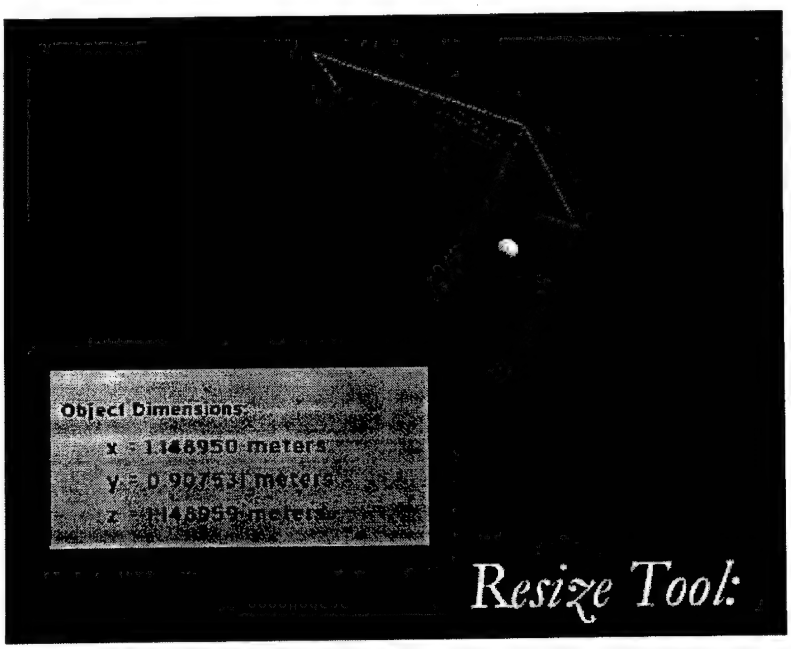



***Image 1.* Shows an object being dynamically resized using the default resize tool.**

The range of properties that can be manipulated is also dynamic, allowing users to specify unique properties that they wish to change for only select types of objects. Currently, the application enables manipulation of both the size and color scheme of an object while the program is running.

[4] 3D Studio Max is a registered trademark of Autodesk, Inc.

In a system that allows for real-time extendibility, it is necessary for applications to make use of dynamic tools, and dynamic menus to interface with these tools. It is imperative for various methods of interacting with the virtual environment to be added as required. This interaction should also be able to take place in a way never imagined by the initial developers of the system, and should not be limited to any basic, underlying, and restricting subsystem.

A method for completing such a task has been implemented for this project. Several tools have been added without restructuring or altering the foundation of the application. An example of this is a grouping tool, which allows users to group together any number of objects. Then, any change made to an object in a group is applied to all other objects in that same group. Such tools are derived off a base tool class, which contains only those properties common to all tools, and a means to interface with e-Touch. They are independent of the foundation application, and are only specific to the very general object type that is used throughout the application. This proves the ability of the application, but more importantly, the new standard of development, to allow for the kind of extendibility warranted and required by such an ambitious project.

### 3.4 Menus

A menu system is another aspect of this project. An important part of this feature is the three dimensional nature of the menus, whose basic implementation is in the e-Touch API. Although visually similar to older windowing systems, giving them a sense of familiarity, these menus are quickly realized to be much more functional than their 2D counterparts. Menu objects, such as buttons and sliders, are also in 3D. Using the PHANToM haptic device, sliders are moved, buttons are clicked, or knobs are turned to adjust aspects of an object. This is a very powerful feature when coupled with a device such as the PHANToM, as buttons and other objects have a physical response when they are activated. Users can feel 3D buttons being pushed, similar to their real world equivalents, such as the buttons on a telephone. A particularly useful example is the 3D slider used for changing the color of an object. A wire frame cube is used to represent all available colors. The slider can be positioned by the user at any point in the cube, and the object's color is determined by the slider's location, see *Image 2.*

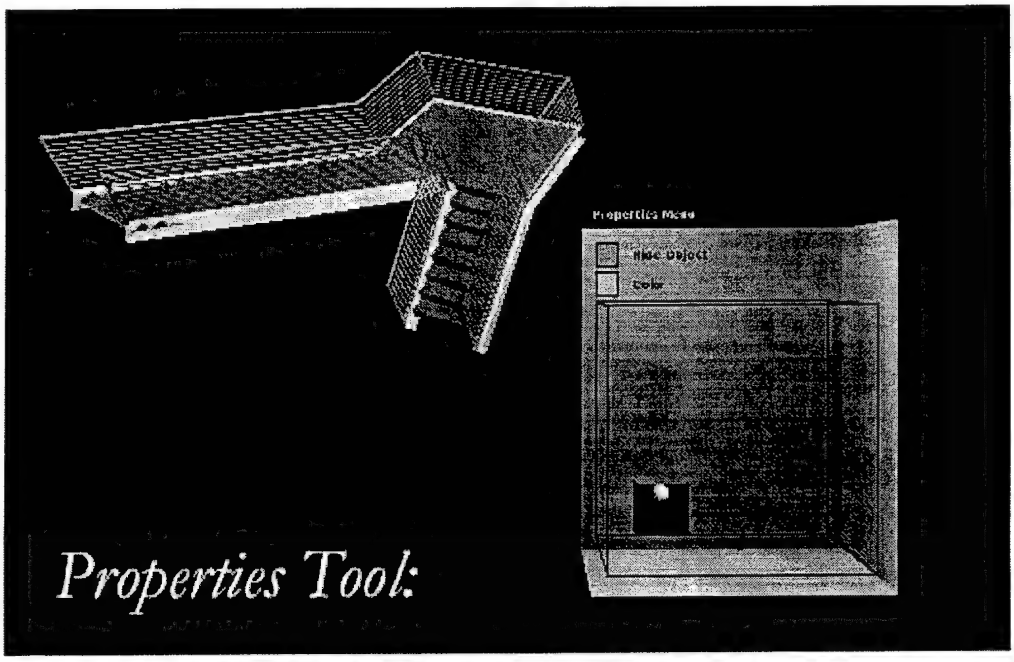



***Image 2.* Shows an object's color property being manipulated using the default properties tool and it's corresponding dynamic menu.**

As our particular application strives to exceed the resources of a single machine, which basically means it needs to be expandable at runtime, a dynamic menuing system was built on top of a set of default, foundation menus. An application area where users are able to easily add a variety of objects to the simulation is needed. This is accomplished by way of a menu system that recognizes the addition of new objects and enables their use. Specifically, a particular folder in the application directory has been set aside for storage of objects the program may use. When a user decides to add an object, this directory is scanned for subfolders, which are presented as buttons on an object adding toolbar. When a button on this toolbar is pressed, a corresponding directory is scanned, and, dynamically, a new object selection menu is built. Users then choose which object they wish to add by pressing any of the buttons on the menu. The menu is destroyed when closed, allowing for new source files for objects to be added to the program directory at runtime. These objects will then be represented in the menus the next time an object is to be added. The menuing system is depicted in *Image 3*. Currently, the project has a dynamic menu that recognizes the addition of new objects using a general haptic object file format and allows the user to use new objects without halting program execution. The goal of dynamic loading without shutting down is aimed toward being able to access features from the internet dynamically.

### 3.5 Lessons Learned

Developing an application that conforms to the new standard for visual and haptic development has not been a small task. Although modular programming languages allow programmers to write code that maintains a structure that can support dynamic loading, there is not currently a good program that allows for dynamic loading of code modules. The most promising is a program called Bamboo [6]. Other options include placing code modules within dll's (dynamic linking libraries). This enables both dynamic linking, adding code at the start of execution, and dynamic loading, adding code during execution.

In order to improve current HCI, we were working with SensAble Technologies' Desktop PHANToM. During our work we discovered several limitations of the current haptic technology. For example, a limited range of motion potentially requires scaling of the device motion. Other recognized haptics needs

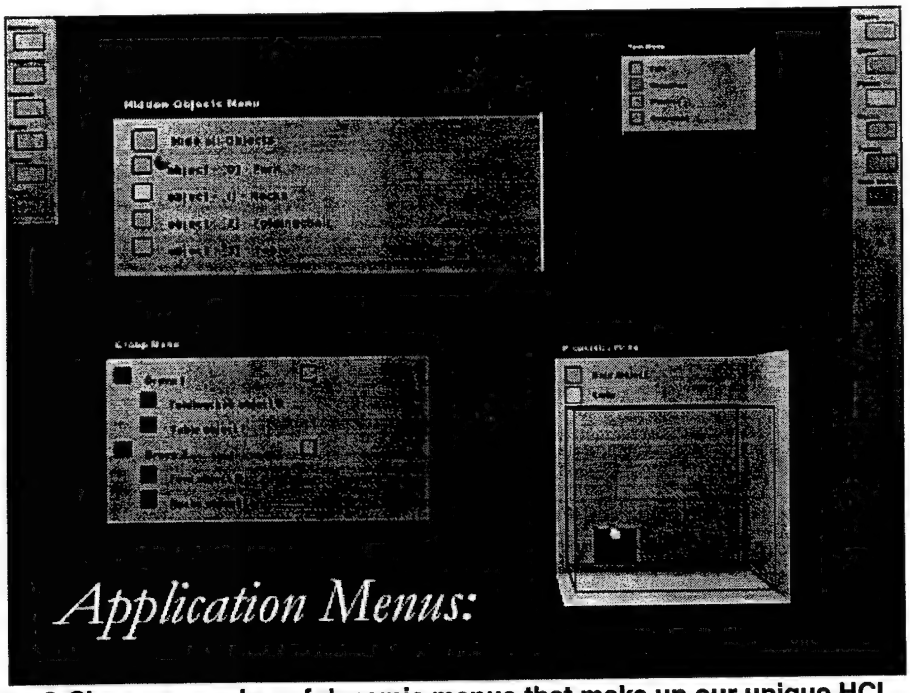



***Image 3.*** **Shows a number of dynamic menus that make up our unique HCI.**

involve a standard format for object property definition. Since graphics and haptics are produced by different output devices, a monitor versus a PHANToM, they have always been treated as two separate entities. Creating a standard format for haptic object information is a necessary step in the development and continuation of the computer haptics field. A behavior constraint library that can be referenced and used to define behaviors and characteristics of specific objects is one promising approach to a common file format. Investigation and communication with the haptics community at large is required for this format to be as inclusive and useful as possible.

Issues to consider regarding supporting multiple users to improve and develop HCI include, but are not limited to, network latency and the effects it has on force feedback[5,6], and platform independence. Haptics is difficult to successfully perform over a network due to the high refresh rates and the subsequent and inherent sensitivity of such simulations. Also, because of the complicated nature of haptics, threads are inexorably an integral part of any simulation. Due to the often platform specific nature of threads, this is a complex and inevitable problem encountered with networking individuals using various system configurations. In our goal for reusable and modular code, platform independence is an important obstacle to overcome.

## 4. Conclusions

Given the evolution of programming languages and techniques, it is possible for more efficient and productive VR applications to be designed. It is now possible to exceed the resources of one's machine and to improve and develop current HCI methods. Specifically, multiple use, multiple user simulations that are composed of dynamically loadable modules, have become a reality and present a bright future for VR development and expansion.

An architectural design application based on and implementing the concepts and ideas presented in this paper is currently under development at Sandia National Laboratories. The foundation of the application has been completed utilizing Novint Technologies' haptics API e-Touch. We have been successful in importing various unique objects from 3D Studio Max files into our simulation, and are able to manipulate their location, orientation, and physical properties dynamically with a number of tools that have been written into the application foundation. The modular, and eventually dynamically loadable, structure of our application has also been verified using several independent tool modules that successfully interact with objects inside the simulation.

The work at Sandia National Laboratories already shows great potential for the full implementation of the ambitious VR standards set forth in this paper. Dynamically expandable, multiple user simulations with a high degree of reusability are on the threshold of reality, and are the future of VR design and development.

## 5. References


[1] Anderson, Tom, "FLIGHT: A 3D Human-Computer Interface and Application Development Environment", Proceedings of the Second PHANTOM Users Group Workshop, Cambridge, MA, 1997

[2] SensAble Devices Inc., "The PHANToM" literature from SensAble Devices Inc. 225 Court St., Vanceburg, KY 41179.

[3] Gutierrez T., Barbero J.I., Aizpitarte M., Carrillo A. R., and Eguidazu A., "Assembly Simulation Through Haptic Virtual Prototypes", Proceedings of the Third PHANTOM Users Group Workshop, Cambridge, MA, 1998

[4] Matsumoto S., Fukuda I., Morino H., Hikichi K., Sezaki K., Yasua Y., "The Influences of Network Issues on Haptic Collaboration in Shared Virtual Environments", Proceedings of the Fifth PHANTOM Users Group Workshop, Cambridge, MA, 2000

[5] Hespanha J., McLaughlin M., Sukhatme G., Akbarian M., Garg R., Zhu W., "Haptic Collaboration over the Internet", Proceedings of the Fifth PHANTOM Users Group Workshop, Cambridge, MA, 2000

[6] Watsen K., Zyda M., "Bamboo- a portable system for dynamically extensible, real-time, networked, virtual environments", Virtual Reality Annual International Symposium. Proceedings, Atlanta, GA, 1998

# The Sound and Touch of Mathematics: a Prototype System

Frances L. Van Scoy, Takamitsu Kawai, Angela Fullmer, Kevin Stamper
West Virginia University
Iwona Wojciechowska - Alderson Broaddus College
Addis Perez, Jesir Vargas - University of Puerto Rico-Rio Piedras
Shelma Martinez - University of Puerto Rico-Mayaguez

The West Virginia Virtual Environments Laboratory is studying ways to use virtual environments technology to assist users with various disabilities. One problem of special interest to us is how to teach mathematics to students with vision disabilities. We are using the PHANToM and sonification techniques to display mathematical functions to visually impaired students.

## Rationale

We agree with Moses and Cobb, "Part of the literacy standard, then, the floor for all students, must be this: when you leave middle school, you are ready to engage with the college preparatory sequence in high school. It's a moving target, but however it's defined, it must then be seen as another floor: when you leave high school, you must be able to engage college curricula in math and science, for full college credit." [1]

National Science Foundation Director Rita Colwell has said, " Every schoolchild must be educated for a productive and contributory place in an advanced information age... K through 12 is the real challenge. As a start, we begin with the assumption that all children can be educated in math and science. This may sound so elementary as to be downright silly! In some places, the educational approach is to sift and sort students early-on. This tells some students right at the starting gate that they can't master science and math -- that we do not expect them to succeed. This becomes a self-fulfilling prophecy, damning to the student and destructive for the country. We must believe in all children so that they learn to believe in themselves..." [2]

Our goal in this project is to help students with disabilities or with different learning styles learn pre-calculus so that college majors in science and engineering are available to them.

## Previous Work

The current project brings together previous work in two areas, use of the PHANToM in displaying map information for pre-trip planning by those with visual impairments [3] and sonification of complex data sets [4-7]. Our sonification work emphasizes the development of algorithms for composing music from complex data sets which will be pleasant to listen to and will assist listeners in discovering data relationships which are not otherwise obvious.

In 2000 we developed a program for the PHANToM which accepts as input a mathematical function of one variable and then constructs a haptic model of a solid block on whose front face a groove representing the function has been carved. The user types on the computer keyboard to enter commands such as those for changing bounds of the domain and range, and the program uses spoken feedback to communicate with the user. A important component of our system is a compiler written using flex and bison which parses the user's function written in Fortran notation [8].

Stephen Brewster's group at the University of Glasgow has built a prototype multimodal haptic math system and done human factors research with both blind and sighted users. In the 2000

version of their system functions were provided by the human factors researchers and hard-coded. They tested for effectiveness of different values of friction and representation of grid lines. They referenced their own work in sonification, writing, "An effective way of presenting the overview of the line graph will shorten the time required in this process and give blind users a better understanding about the graph. Using non-speech sound to provide this kind of quick overview [is] being investigated." [9]

## Specification of New Prototypes

In 2001 we extended our prototype haptic math system to add:

(1) improvement of the user interface,

(2) display of functions of two variables, and

(3) sonification of functions of one variable

## Improving User Interface

The existing version of haptic math provided only the basic functions needed to allow the user to enter a function and change some parameters. All input was done via the keyboard and made several assumptions to simplify the initial interface. Work this summer focused on identifying components required and/or useful in a more complete user interface. This includes not only the listing of these components but consideration as to how best to implement each of them to allow the greatest adaptability for users with varying disabilities or learning styles.

There are two main parts of the user interface: the informational display and the system controls.

The display part of the interface is concerned with how the function is represented to the user. This includes the graphic representation as drawn on the screen, the haptic representation as felt via the PHANToM, and the production of any auditory information. In regards to the multi-modal 'display' there were several issues considered. These included investigation into the way that the curve is drawn to provide a more uniform groove width at points with steep or zero slope, methods of informing the user (other than visually) of discontinuities in the graphed function and providing the user with alternative methods to explore the graph.

The system control part of the interface is concerned with how the user sets parameters, enters functions and in general conveys information to the system. For the system control part some issues considered include variations in the format of the functions that may be entered, the means of detecting and informing the user of errors in the function entered, the provision of user access to parameters and the ability of the user to selectively alter parts of the multi-modal display to adapt to individual needs. The existing version restricts the choices for system control keys to those on the left hand. This is an unnecessary and potentially problematic restriction for several reasons. It assumes a right-handed user. It assumes the traditional 'qwerty' keyboard, a problem since other keyboards are used by potential users. Also, it forces some command sequences to be rather arbitrary. The restriction is unnecessary as it is not possible on a qwerty keyboard to use only left hand sequences to enter characters such as the parentheses needed in the functions entered. This also forces the user to memorize the correspondence between keys and parameters - a potential problem for a novice user who may not even realize what parameters are available. It would also be desirable for the user to be able to alter the display. For example, a user might find the musical representation of the curve distracting and choose to turn it off.

## Display of Functions of Two Variables

We also developed a new module for displaying functions of two variables.

First, the visual representation had to be changed. The original one-variable version carves a groove in a block of wood, giving a 3-d representation of a 2-d curve. In the case of a function of two variables, we are dealing with a 3-d surface. To obtain this surface, the plane (x,y) is triangulated and the z-value of the function is calculated for each triangulation vertex. This produces the set of 3-d points (x,y,x) that are used to create the tripoly mesh that PHANToM user can "feel." To make the surface easier to explore without falling off the surface, we added a bounding box.

The second change involved modifying the function parser to accommodate functions of two variables. This major modification required making changes to the interface between the display module and the parser. Additional arguments were needed. Also the type and meaning of one of the parameters was changed. The latter change was necessary, because of the precautions taken in the previous version against mathematical roundup errors.

Work remains to be done. Currently the function surface is displayed with the (x,y) plane horizontal. Additional display options can be added such as displaying the plane vertically or allowing the user to rotate the surface. Also for sighted users the color of the display should be changed and lighting added to create shadows. An artifact of the 2000 system is that the expression defining the function is re-parsed for every pair (x,y), with a major loss of efficiency. We need to restructure the system so that the function is parsed one time and then evaluated once for each (x,y) pair.

## Sonification of Functions of One Variable

In sonifying functions of one variable we map a function to a line of music, mapping x-coordinates to time and y-coordinates to pitch. We do this in order to give the student a way of remembering the shape of a function in the belief that many students will be better able to remember a melody describing the sine function rather than only the muscular sensations of tracing it. We use two different encodings--one using the western chromatic scale and one using micropitch--in the belief that melodies based on the familiar scale will be easier to remember but that micropitch will give a more accurate representation of a function (since the chromatic scale encoding essentially represents continuous functions as step functions).

Figure 1 shows an example illustrating how we sonify functions of one variable. The music is that generated for $f(x) = \tan(x)$, for $0 \leq x \leq 2\pi$.

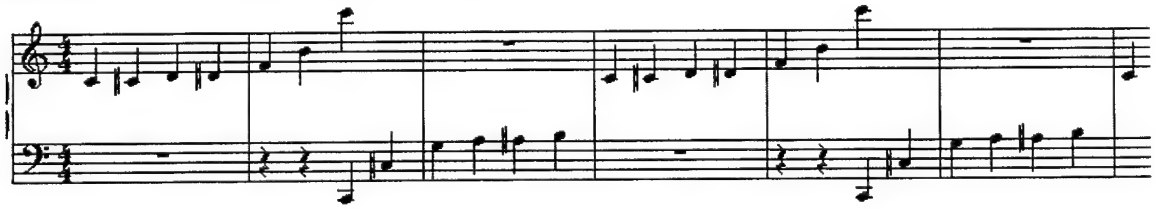


Figure 1. Sonification of Tangent Function

## Continuing Work

We are now designing a human factors study for testing this PHANToM-based prototype with secondary school students and doing preliminary design for some course modules incorporating this approach.

## References

[1] Moses, Robert P. and Charles E. Cobb, *Radical equations: math literacy and civil rights*, 2001, Boston: Beacon Press, p. 16.

[2] Colwell, Rita, address to DC Science Writers Association, September 8, 1998, available at http://www.nsf.gov/od/lpa/forum/colwell/rc80908.htm

[3] Van Scoy, Frances L. Vic Baker, Chaim Gingold, Eric Martino, Darrin Burton, "Mobility Training using a Haptic Interface: Initial Plans," in Salisbury, JK and Srinivasan, MA (Eds), "Proceedings of the Fourth PHANTOM Users Group Workshop," AI Lab Technical Report No. 1675 and RLE Technical Report No. 633, MIT, November 1999, available at http://www.sensable.com/haptics/community/pdf/PUG1999.PDF

[4] Van Scoy, Frances L., "Sonification of Complex Data Sets: An Example from Basketball," Proceedings of VSMM99 (Virtual Systems and MultiMedia), Dundee, Scotland, September 1-3, 1999.

[5] Van Scoy, Frances L., "Sonification of Remote Sensing Data: Initial Experiment," Information Visualisation 2000, London, UK, July 19-21, 2000

[6] Mutnuri, Sunitha, "Sonification of GIS Data," Master of Science in Computer Science problem report, Lane Department of Computer Science and Electrical Engineering, West Virginia University, December 2000.

[7] Wallace, Carl, "Sonification of Basketball Data," Master of Science in Computer Science problem report, Lane Department of Computer Science and Electrical Engineering, West Virginia University, December 2001.

[8] Van Scoy , Frances L., Takamitsu Kawai, Marjorie Darrah, Connie Darrah, "Haptic Display of Mathematical Functions for Teaching Students with Vision Disabilities: Design and Proof of Concept," In *Proceedings of the First Workshop on Haptic Human-Computer Interaction*, Glasgow, Scotland, August 31-September 1, 2000, available at http://www.dcs.gla.ac.uk/%7Estephen/workshops/haptic/papers/haptics_and_hci_workshop.zip

[9] Yu, W., R. Ramloll, and S. A. Brewster (2000). "Haptic graphs for blind computer users," in *Proceedings of the First Workshop on Haptic Human-Computer Interaction*, Glasgow, Scotland, August 31-September 1, 2000, available at http://www.dcs.gla.ac.uk/~stephen/papers/Assets2000.pdf

## Acknowledgements

This work is being conducted in the West Virginia Virtual Environments Laboratory in the Lane Department of Computer Science and Electrical Engineering of West Virginia University and is supported by the EPSCoR programs of the National Science Foundation and the state of West Virginia and by the NSF CISE Research Experiences for Undergraduates program.

Darrin Burton of the WVU Center for Assistive Technology has provided valuable advice on usability of the system.

Nicole Johnson, a WVU McNair scholar, and Yusuke Abe, a Fairmont State College student, were active participants in VEL discussions about this project.

# Fast Haptic Rendering of Complex Objects Using Subdivision Surfaces

Chris Raymaekers Koen Beets Frank Van Reeth

Expertise Centre for Digital Media, Limburg University Centre
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{*chris.raymaekers, koen.beets, frank.vanreeth*}*@luc.ac.be*

**Abstract**

Haptic rendering of meshes of arbitrary topology is a difficult and time-consuming process. This paper presents the use of subdivision surfaces in order to solve this problem. An overview of subdivision surfaces is given and the algorithms needed to calculate the surface contact point are discussed. We will show that this algorithm is faster than traditional algorithms.

## Introduction and Related Work

Haptic rendering of complex objects needs a lot of calculating power. Even with modern computers, limiting the number of calculations that need to be made on a complex object can be very difficult. Two approaches are frequently used: mathematical representations of the objects, and polygonal models. The GHOST SDK (SensAble, 2001) uses both methods: a mathematical representation is used for simple shapes, such as cubes and cones, while arbitrary meshes are represented by a polygonal model.

Most of the time, only a limited number of polygons is used in order to keep the calculations within the 1ms interval of the haptics loop. However, this also limits the precision of the model. An alternative is the use of mathematical representations, such as NURBS. This however introduces other problems: representing models of arbitrary topology is for instance very difficult.

In our research, we would also like to deform the objects. Using NURBS, the seams of the patchwork can become visible during deformation. This problem also occurs in computer animation and has been solved by using subdivision surfaces. A famous example is the Pixar movie "Geri's Game" (DeRose et al., 1999).

## Subdivision Surfaces

A subdivision surface is a surface that is defined as the limit of a series of refinements $M_1, M_2, \ldots$, starting from an original control mesh $M_0$. Because of this property, they support level-of-detail. Since they can also be used to efficiently represent objects of arbitrary topology, and they can be modified easily to support features such as creases and boundaries, it is no surprise subdivision surfaces are already used in computer graphics and computer animation. Figure 1 shows a control mesh and the first refinement.

A wide variety of subdivision schemes for surfaces exist, with an equally large variety in properties. Two of the most well-known subdivision schemes for surfaces are the Catmull-Clark scheme, which works on quadrilateral meshes, as described in (Catmull and Clark, 1978), and the Loop scheme, which is triangular (Loop, 1987). The types of surfaces generated by these schemes differ, but the general principles of subdivision surfaces remain the same. In our research, we use the triangular loop scheme because triangular polygons are very well suited for modelling freeform surfaces, and they can be easily connected to an arbitrary configuration . For a detailed explanation of subdivision and subdivision surfaces, we refer the interested reader to (Zorin and Schröder, 2000).

The Loop scheme, based on the three-dimensional box spline, is an approximating face-split scheme for triangular meshes, invented by Charles Loop. The resulting surfaces are $C^2$ everywhere except at extraordinary vertices — with a valence different from 6 — where they are $C^1$.

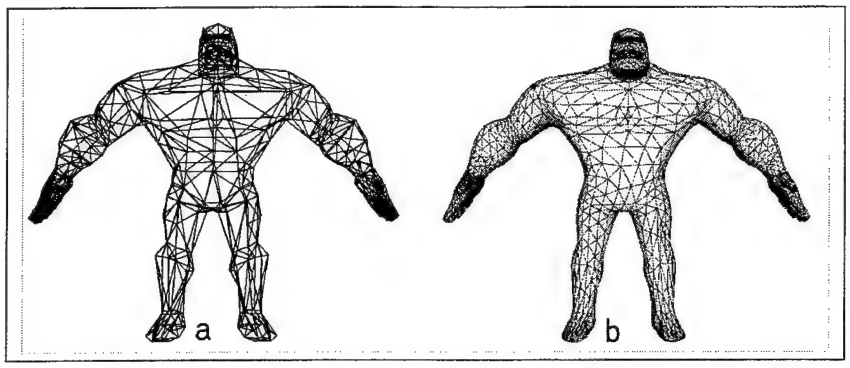



Figure 1: Loop control mesh and first refinement

An iteration of the scheme consists of two stages. In the first stage, known as the splitting stage, a new vertex is added in the middle of each edge, and both the old and new vertices are connected to form 4 new triangles for each old triangle. In the smoothing stage, all vertices are averaged with their surrounding vertices. This smoothing step, together with the weights used, is visualized in figure 2. Loop's original choice for $\beta$ was $\beta = \frac{1}{k}(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}cos(\frac{2\pi}{k})^2)$, where $k$ is the valence of the central vertex, but other choices are possible as well (Warren, 1995). Figure 2 also shows that the support — which is the region over which a point influences the shape of the limit surface — of the Loop scheme is small and limited.

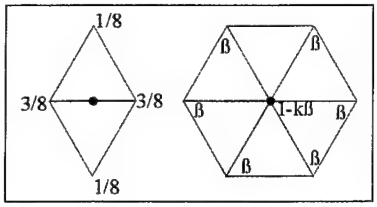



Figure 2: Subdivision Mask of the Loop Scheme

## Calculations using Loop Surfaces

During haptic rendering of polygon meshes, all of the object's faces generally need to be processed. This is no longer necessary with subdivision surfaces. The multiresolution properties of subdivision surfaces can be exploited so that only the control mesh has to be processed as a whole. In the processing stages of the following, more detailed, subdivision levels, the results of the previous test are used, thus leading to a smaller number of polygons that have to be processed. This leads to a huge increase in speed.

As mentioned in the previous section, a subdivision surface is defined as the limit of a series of surfaces. This gives rise to two interesting properties:

- Every face at level $n-1$ can be linked to four faces at subdivision level $n$.
- As can be seen from figure 2, the new co-ordinates of a vertex are influenced by the surrounding vertices. Using the loop subdivision scheme, most vertices have a valence of 6, so 6 vertices are needed to calculate the new co-ordinates. Generalizing this for a triangle (figure 3), each vertex of the triangle is influenced by its 6 neighboring triangles. Since a number of these neighboring triangles are shared, 12 neighboring triangles are needed to calculate the new co-ordinates of each of the triangle's vertices. The grey triangle in figure 3 is the triangle that is subdivided.

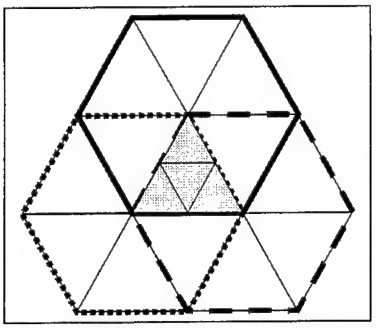


Figure 3: Area which influences a single triangle

The next two sections explain the 2 problems that need to be solved. In both cases the algorithm starts from the control mesh at level 0, leading to an accuracy up to an arbitrary subdivision level $n$, which contains $4^n$ times as much triangles as level 0 does.

## Performing the inside-outside test

Consider a control mesh $M_0$, consisting of $a$ triangles. The following steps describe the algorithm that checks wether a point $p$ lies inside or outside the object.

1. Shoot a semi-infinite ray, starting at point $p$.
2. Select all the intersected faces and the face closest to the point being tested.
3. Extend this selection by including all triangles in the 1-neighborhoods of all vertices of the intersected faces.
4. Replace all the selected faces by their subdivided children. The number of triangles is multiplied by 4. Again test all triangles in this selection for intersection with the ray.
5. Check wether the number of intersections found is different from the previous number. This can happen because the refined meshes "shrink" in areas where the control mesh is convex. In concave regions, the refined meshes grow outside of the control mesh.

   If the number has changed, go to step 7, otherwise proceed with step 6.
6. If the required subdivision level is not yet reached, go to step 2.
7. If the number of intersections is odd, the point lies inside the polymesh. Otherwise it lies outside the polymesh.

## Calculating the SCP

Using the results of the previous calculations, the SCP can be calculated.

1. If in the previous algorithm the required subdivion level is found go directly to step 7.
2. Select those faces from the selected faces of the current subdivision level which are closest to the point being tested.
3. Extend the selection to faces which contain vertices in the 1-neighborhood of all vertices in the selected faces.
4. Replace the selection with its next subdivision level.

5. Select the closest face from the previous selection.
6. If the required subdivision level is not yet reached, go to step 3.
7. Calculate the exact intersection point of the closest face. This is the SCP.
8. For each vertex of the triangle the limit normal vector is calculated by taking the vector product of the following two vectors.

$$t_1 = \sum_{i=0}^{k-1} \cos\frac{2\pi i}{k} p_i \qquad t_2 = \sum_{i=0}^{k-1} \sin\frac{2\pi i}{k} p_i$$

The normal in the SCP is calculated by interpolating these three normals.

## Comparison

Using subdivision surfaces, only a small number of triangles need to be processed. Consider again the control mesh $M_0$, consisting of $a$ triangles. At level 0 $a$ intersection tests have to be performed. If $k$ intersections are found ($k$ is typically a very small number), in each step these triangles and their neighboring triangles need to be subdivided. In the worst case scenario, where the neighboring zones are al disjunct, $k*(1+12)*4 = k*52$ intersection tests have to be performed for each subdivision level. The maximum number of tests (if the test is not successful before reaching level n and all zones are always disjunct) is $a + n * k * 52$ (please note that the subdivision process is performed in preprocessing stage) . If a "normal" polymesh with the same number of polygons has to be checked, $a * 4^n$ triangles would have to be checked.

For instance, suppose a control mesh, consisting of 100 triangles, is checked until level 4 (a typical level) and 5 intersections are found. This leads to $100 + 4 * 5 * 52 = 1140$ checks. The equivalent polymesh consists of $a * 4^4 = 25600$ triangles which leads to more than 20 times as much checks.

When using adaptive subdivision, where the fact if a triangle is subdivided depends on the surface area of the triangle (compared to the other triangles of the mesh), even a smaller number of tests is needed.

## Conclusions

We proposed a technique for fast haptic rendering by using subdivision surfaces, which can greatly improve speed. Also, by using subdivision surfaces, we can evaluate objects of any topology up to an arbitrary refinement level.

## Acknowledgments

Part of the work presented in this paper has been funded by the Flemish Government and EFRO (European Fund for Regional Development).

## References

Catmull, E. and Clark, J. (1978). Recursively generated b-spline surfaces on arbitrary topological meshes. *CAD*, 10(6):350–355.

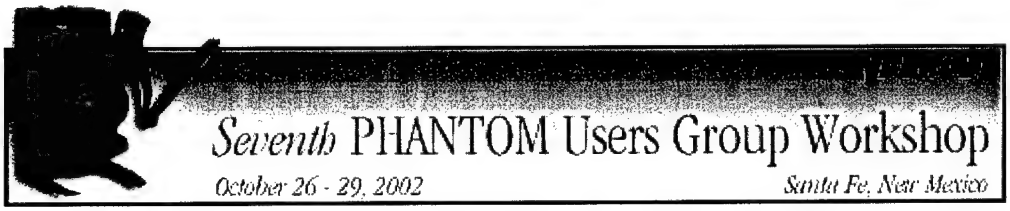
DeRose, T., Kass, M., and Truong, T. (1999). Subdivision surfaces in character animation. In *Proceedings of the SIGGRAPH 1999 annual conference on Computer graphics*, pages 85–94, Los Angeles, CA, USA.

Loop, C. (1987). Smooth subdivision surfaces based on triangles. Department of mathematics, University of Utah, Utah, USA.

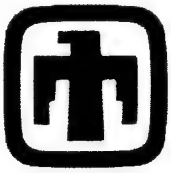
SensAble (1996–2001). *GHOST Programmers Guide.*

Warren, J. (1995). Subdivision methods for geometric design. Rice University.

Zorin, D. and Schröder, P. (2000). *Subdivision for Modeling and Animation.* Number 23 in Course Notes for SIGGRAPH 2000. ACM.




# PAPER PROCEEDINGS

Sandia National Laboratories

hosts the
*Seventh* PHANTOM Users
Group Workshop
in beautiful Santa Fe, New Mexico

Sponsors:

Telemedicine & Advanced Technology Research Center
of the United States Army Medical Research and Materiel Command
Novint Technologies, Inc., Albuquerque, NM
SensAble Technologies, Inc., Woburn, MA

# PUG 2002 Symposium Agenda

**SUNDAY**

| | |
|---|---|
| 8:30-9:00 AM | Opening Remarks -- Conference Chair(s) |
| 9:00-10:00 AM | Keynote Speaker – Francois Conti, Force Dimension<br>Current Status of Haptics: Part I |
| 10:00-10:15 AM | Special Report I – Ghost Release on Linux |
| 10:15-10:30 AM | Break |
| 10:30-12:00 PM | Paper Session I – Applications |

1. Implementing Haptic Feedback in a Projection Screen Virtual Environment – Andrew Fischer and Judy Vance, Iowa State University
2. Haptic Interaction with 3D Ultrasound Data: The e-Touch Sono System – Walter Aviles, Tom Anderson, Novint Technologies, Albuquerque, NM

| | |
|---|---|
| 12 Noon - 1:00 PM | Lunch |
| 12 Noon - 2:00 PM | Demo Room Open<br>Enjoy Bishop's Lodge/New Mexico Siesta |
| 2:00 – 3:00 PM | Paper Session II – User Interfaces |

1. A 3-D Lasso Tool for Editing 3-D Objects: Implemented Using a Haptics Device – Marjorie Darrah, Institute for Scientific Research, Fairmont, WV, Alicia Kime, School of Science and Mathematics, Frances Van Scoy, Lane Department of Computer Science and Electrical Engineering.
2. Touch & Tell: A game-based tool for learning to use the PHANToM – Eric Acosta, Bharti Temkin PhD, Texas Tech University

| | |
|---|---|
| 3:00 - 3:15 PM | Special Report II – Computer Haptics DVD 2002 |
| 3:15 – 3:30 PM | Break |
| 3:30 PM – 4:30 PM | Paper Session III – Performance |

1. Accessing the increase in haptic load when using a dual PHANToM setup – Joan De Boeck, Chris Raymaekers, Karin Coninx, Expertise Centre for Digital Media, Limburg University Centre, Belgium
2. Response Time Consistency of the GHOST force loop – A.E. Kirkpatrick and Jason Sze, Simon Fraser University, Canada

| | |
|---|---|
| 4:30 – 5:00 PM | Special Report III – Sensable Technologies, Inc. |
| 6:00 – 8:30 PM | Dinner with "Touching New Mexico" Presentation |
| 6:00 – 10:00 PM | Demo Room Open |

## Keynote Speaker -- Francois Conti

Co-Founder, and Co-CEO, Force Dimension
Francois Conti received his MS in Electrical Engineering from The Swiss Federal Institute of Technology in Lausanne (EPFL). He joined the VRAI group in 1998 and was responsible for the development of an endoscopic surgery simulator, today commercialized by Xitact. In 1999, He participated in the designed of the DELTA Haptic Device and co-founded Force Dimension in 2000. In 2001, he joined the Robotics Laboratory at Stanford University where he is currently pursuing his Ph. D. in the field of soft tissue modeling and simulation for real-time applications.

## Force Dimension

Force Dimension, a spin-off of EPFL founded in 2000, designs and manufactures high-end force feedback interfaces for research and industrial applications. Force Dimension has been active in the automotive and aerospace industry for augmented haptic teleoperation and telemanipulation applications.

Company Milestones:

| | |
|---|---|
| 2000 | Force Dimension is founded and the 3-DOF DELTA Haptic Device is commercialized. |
| 2001 | The 6-DOF DELTA is introduced. |
| 2001 | Support of Novint's eTouch API. |
| 2002 | The Mini DELTA is presented at Eurohaptics for the first time. |
| 2002 | Support of the ReachIn API. |
| 2002 | NanoFeel, a spin-off company from EPFL, is launched to commercialize the NanoManipulator: a force feedback and visual software interface, based on the DELTA Haptic Device, controlling in real-time an Atomic Force Microscope (AFM) |

# Porting the GHOST SDK to Linux

**Francis D. Bogsanyi**
**Walter A. Aviles**
Novint Technologies Incorporated
http://www.novint.com

## Overview

The GHOST® SDK (General Haptic Open Software Toolkit) is a widely used commercial C++ haptic rendering toolkit [1]. It is aimed at providing a uniform interface to the entire SensAble Technologies PHANTOM™ line of haptic interface devices and handling the core haptic rendering components for tools and applications that utilize these systems.

GHOST handles the initialization, low-level device safety and communication components of haptically enabled tools and applications. It also provides a core 1-kHz servo loop and a simple, real-time, two-thread (i.e., application and haptic threads) model for application developers to use in their efforts. In addition to providing a library of prismatic objects, polygonal geometry renderer/reader and haptic effects, GHOST also provides application developers with the ability to create their own force feedback algorithms. With Version 4.0 of the GHOST SDK, and the introduction of the gstDeviceIO class, developers and researchers can also access the haptic interface device at a sensor/actuator level and control the workings of the servo loop for more fundamental haptic research and development.

In order to help meet a long-standing need in the Linux community for haptic device support, we undertook a port of Version 4.0 of the GHOST SDK to the Linux operating system. The remainder of this paper discusses some of the key issues that were encountered and design decisions that were made in developing the Linux version of GHOST. These factors are not unique to this effort. They affect any Linux software where real-time performance is important, support is a concern, device drivers must be developed or where complete source code is not available or cannot be released.

## Linux Port Philosophy

Several principles guided the port of the GHOST SDK to Linux. First of all, the API of the GHOST SDK should not change for the Linux port. The API should be identical to that of the Windows™ version of GHOST. Second, ease of support dictated that a single, widely available and consistent version of Linux was to be used for the port. The distribution chosen was RedHat™ Linux release 7.2. The Linux port should also have minimal dependence on additional packages. Finally, the port was intended to have a minimal impact on the GHOST "core" code-base. These principles were not, in general, firm requirements and some of the challenges encountered in meeting these principles and the tradeoffs that were taken are discussed below.

# Challenges

Linux, in its purest form as an operating system, kernel, and philosophy, is not an ideal platform for haptic software and poses a number of challenges for commercial and C++ software development. Linux is not well suited for supporting real-time applications, it is not a stable platform for professional software development and it does not support binary distribution of drivers.

## Real-Time Support

Linux is not a real-time kernel. Although real-time sub-kernels and extensions for Linux are available, these pose challenges of their own. There has been some volatility in the real-time Linux market and many of these extensions and adaptations are either not maintained, unsupported, or proprietary. Our first principle and the desire to reach the broadest audience cautioned against the use of proprietary or unsupported kernels.

GHOST has a soft real-time requirement for the servo loop, which should execute at approximately 1-kHz. Additionally, it has a two-thread model that should be supported on single-CPU workstations with guaranteed progression of the application thread. An operating system kernel offering sub-millisecond timing accuracy, "real-time" priority thread scheduling, and suitable thread sleep functionality, can meet these requirements. While the standard Linux kernel can meet the first two requirements, it lacks a suitable thread sleep facility. Of the myriad mechanisms for sleeping, such as usleep(), sleep(), and itimers, only nanosleep() supports sub-millisecond sleeps. Unfortunately it accomplishes this through a busy-wait, which doesn't permit progress of the application thread.

## Platform Stability

Linux is not a stable platform for professional development and makes it challenging to provide support for any software application. The software development model used for Linux, and described in [2], offers benefits such as rapid code development, adaptation and improvement, fast identification and elimination of bugs, and competition. In short, it accelerates software evolution. The pace of advance has the potential to greatly aid end users, system administrators and software developers. Security holes are quickly identified and repaired, new capabilities are added, and the software simply gets better faster. Unfortunately, this accelerated evolution has a significant downside for commercial software development. Commercial software relies on a stable underlying platform and most companies can support only a relatively limited range of software and hardware configurations. Linux does not provide this stability. For example, each of the releases of RedHat Linux in 2001 and 2002 has provided incompatible versions of the g++ compiler, the Standard C++ Library and the X-Window system. Additionally, some have shipped without an implementation of Motif and others have provided 2 or 3 incompatible implementations. Multiple incompatible binary kernels were supplied with each release of the operating system, and updates were provided between releases that were also incompatible at the source-level. Each of these packages plays a significant role in the GHOST SDK, and the incompatible C++ ABIs (i.e., Application Binary Interfaces) provided by these operating system releases are particularly troublesome.

## Binary Drivers

By design, Linux kernels are explicitly not binary compatible. Binary distribution of drivers is actively discouraged. Configuration options, processor type and version, kernel version, patches, compiler version and build environment all affect the kernel's binary interface and conspire against the release of binary driver modules. RedHat 7.2 alone offers a bewildering variety of binary kernels, all of which are binary incompatible. A widely used binary driver module for NVIDIA™ graphics cards [3] illustrates the problem, with 44 versions supplied at the time of writing, including 6 for RedHat 7.2 and 14 for RedHat 7.3. All of this makes support of binary drivers extremely difficult. It also presents challenges to developers who desire to avoid distributing source to drivers due to intellectual property, safety or legal liability concerns.

# Solutions and Compromises

It has oft been stated, "there is no such thing as portable software, only ported software". Ports of the GHOST SDK exist for the SGI IRIX™, Sun Solaris™ and Microsoft Windows™ operating systems. The code has successfully passed through three C++ compilers; driver examples exist for three operating systems; and "two and a half" operating system APIs are supported. Linux shares the Posix API with IRIX and Solaris, but like these two UNIX variants, an SDK of the complexity of GHOST requires non-standard extensions. Much of the port, therefore, was straightforward selection of code used by the two UNIX ports, while the remainder required identification and use of "similar" proprietary extensions, and resolving compilation issues.

## Real-Time Approach

Linux is a "UNIX-like" operating system, implementing various Posix APIs. An incomplete Posix Threads implementation existed at the time of the port and this formed the basis for the two-thread model. Limitations of the Posix Threads implementation include the lack of a high-resolution timer or sleep facility, and the incorrect operation of the exit() system call. Posix "threads" are implemented under Linux as operating system processes that share some kernel-level process data, such as file descriptors and page tables. The kernel is largely unaware of the existence of threads. In particular, threads do not share a process ID and the exit() system call does not stop all threads in the "process". The correct behavior can't be implemented without kernel support and a "belt and suspenders" approach had to be implemented to try to ensure correct shutdown of the SDK and a clean hardware state. Nonetheless, it is still possible for the SDK to be "caught with its pants down" under certain circumstances. Recent versions of the kernel and Posix Threads library have corrected these problems, but these versions are not generally available in operating system distributions like RedHat.

Hardware assistance was required to alleviate the lack of suitable timer facilities. The PHANTOM PCI interface card and the control circuitry in the parallel port PHANTOM models provide an interrupt signal. We programmed the hardware to issue an interrupt signal at a 1 kHz frequency. The drivers supported the poll() system call that was used by the real-time thread to delay execution and free the processor for other tasks, while still guaranteeing a fast wakeup and evaluation of the servo loop at 1 kHz. The poll() call returned when a hardware interrupt was received. To guarantee timely reclamation of the processor by the real-time thread, the kernel-space interrupt handlers triggered a scheduling operation and the real-time thread was given the highest priority. It is not possible to provide a 1 kHz servo loop while guaranteeing progression of other threads on a single processor system under Linux without this hardware support. Additionally, it is desirable to avoid delays due to dynamic memory paging, and this is

accomplished with the mlockall() system call. Unfortunately, this can only be called by the target process and requires super-user privileges. Hence all GHOST-based applications must be executed by root or be setuid root. The IRIX port didn't attempt to lock memory pages and used a setuid root utility program to boost the priority of the real-time thread. It is significantly more common for software developers to have super-user privileges on their Linux workstations and we regarded this requirement as a suitable compromise.

## Platform Stability Work Arounds

Two implementations of the Motif user interface toolkit are in common use in the Linux community. Lesstif is an independently developed clone, while OpenMotif is an Open Source release of the standard OSF code-base. Lesstif shipped with RedHat 7.2, but lacked a ComboBox widget compatible with that used by the GHOST SDK. It was necessary to obtain the necessary widget and link it into the GHOST libraries. The Lesstif library offers two "personalities": Motif 1.2 and Motif 2.1. GHOST is built against Motif 1.2. GHOST additionally depends on an OpenGL rendering area widget suited for use with Motif. Unfortunately, while a suitable widget library is part of the Mesa toolkit, RedHat doesn't supply it. We resolved this by building the relevant library from the source package supplied by RedHat and shipping this as a separate package.

RedHat 7.2 provided version 2.96 of the g++ compiler and the GNU Standard C++ Library. Although more recent versions of the compiler and library are significantly closer to fully implementing the C++ standard, the versions used have some limitations. These limitations weren't a significant issue for the port, but as these tools have improved, their interpretation of the standard has become stricter. Language constructs such as passing anonymous local variables to functions expecting non-const reference parameters are accepted by the more permissive SGI, Sun and Microsoft compilers, but were correctly rejected by the g++ compiler. It is likely that attempts to compile the SDK with more recent tools will uncover similar errors.

A more significant problem is the evolution of the C++ Application Binary Interface provided by the g++ compiler. Each new release of the compiler, since version 2.95, has broken backwards compatibility with the previous versions. While we could have chosen to adopt the potentially more advanced release 3.1 of the compiler and release 3 of the standard C++ library, which were available at the time of porting, this would have presented difficulties for developers wishing to use GHOST with other C++ APIs and would have required significant upgrades to the operating system, in opposition to our guiding principles. We instead chose to use only the tools available in RedHat 7.2, at the expense of tying the port to that platform and requiring additional porting and support effort, should demand arise for use of the SDK on more recent releases or other Linux distributions.

## Driver Distribution Approach

A large part of GHOST's responsibility is communication with the PHANTOM devices. This naturally requires some kind of device driver support. While several device interfaces have been previously used, only the parallel port interface used by the PHANTOM 1.5 6-DOF and the PHANTOM Desktop is considered "current", and the PCI interface card used by other PHANTOM models is regarded as a useful "legacy" interface. Others, including ISA and VME-based cards, were not included in the port. The IO library therefore needed to support two interface models. The Linux kernel includes a driver offering user-space access to low-level

functionality of the parallel port. This was sufficient to support communication between the PHANTOM and user-space applications and was used as-is. No such facility was available for the PCI interface, however, and it was necessary to implement a kernel driver module for communication with the PCI card. The software model for the parallel port interface streamed data in IEEE-1284 EPP mode and viewed the hardware registers in the PHANTOM control card as a block of memory addressed through the parallel port address register and read and written as a series of bytes through the parallel port data register. A similar model was adopted for the PCI driver interface, except that ioctls were provided for directly reading and writing registers individually or in blocks.

Providing a kernel driver module for PCI support required facing the lack of binary compatibility in the Linux kernel ABI. The full source code for a sophisticated kernel driver module could not be released due to intellectual property and safety concerns (i.e., part of the PHANTOM haptic interface's device and user safety system is in software and should not be circumvented). The compromise reached was to provide the register-addressing model described in the previous paragraph and to maintain sequencing logic and register addresses in binary-only user-space code. The source code for the kernel driver module could therefore be provided to customers. Due to the legacy status of the PCI interface card, the wide variety of kernels provided by RedHat in the 7.2 release and its updates, and the propensity of Linux users to compile their own kernels, we chose to ship the kernel module as source-code only.

## Remaining Issues

We have successfully ported the GHOST SDK to one release of one distribution of the Linux operating system. At the time of writing, this port is only a couple of months old, yet it depends on a specific version of the RedHat Linux operating system that is two complete releases out of date. We are by no means unique in this respect, but it is worth asking whether we could do better.

RedHat Linux 8.0 is shipped with version 3.2 of the g++ compiler. This release promises enhanced forward binary compatibility of the C++ ABI, although it breaks backwards compatibility. This is a significant incentive to update the GHOST port for RedHat 8.0, as this could potentially allow use of the SDK on releases of Linux from other vendors providing the same or a newer release of the compiler. It is not known whether the Mesa/Motif compatibility issue remains.

The Linux kernel evolves quickly. The GHOST SDK port is largely dependent on two kernel device drivers: the parallel port driver, which ships with Linux; and the PHANTOM PCI card driver, which ships with GHOST. The parallel port driver provides a relatively stable ABI to user-space applications through the Linux ioctl mechanism. A number of higher-level drivers are supported for different types of attached devices, such as printers, storage devices, and the PLIP transport protocol. While we could have chosen to implement our own higher-level driver to support the PHANTOM, this would have required writing and maintaining yet another kernel module. The alternative we selected, using the low-level parallel port interface provided by the parport driver module, is likely to be more stable in the short term, but remains subject to the whims of the device driver implementers. In particular, we rely on prompt delivery of hardware timer interrupts and resumption of the real-time thread. The internal kernel APIs for PCI drivers have evolved considerably over the life of the kernel and that evolution continues. Supplying a source-code driver for the PCI interface card shields us from binary compatibility issues, but is still sensitive to changes in the internal kernel APIs. This problem has already affected the driver,

with the addition of the no_llseek() function and the MODULE_LICENSE() macro in version 2.4.9 of the kernel, which was released as an update to RedHat 7.2 while the port was in progress.

The current error reporting facility is inadequate. Reporting errors on the standard error file stream and requiring user input is problematic for graphical applications, which may be started without or may obscure a terminal window. The observable effect is that the application stops responding, or the graphical representation of the PHANTOM stops moving, and the cause and solution are not always immediately obvious, even for experienced users such as the authors! The present solution was chosen due to the complexity of detecting whether X or some other graphical interface is available and providing suitable reporting under all conditions. Programmatic error reporting and response would alleviate this problem.

The PHANTOM Mouse facility was not ported to Linux. Porting the facility would add significant complexity to supporting the SDK. A binary module needs to be written for the X server and, beside the source-level incompatibility with similar modules for IRIX and Solaris, this suffers from exactly the same problems as binary kernel modules. A binary module for the X server may only be used with the specific server binary it was built against.

# Conclusion

Version 4.0 of the GHOST SDK was successfully ported to Linux. This operating system is particularly challenging for software development efforts where real-time operation, software support or binary distribution are important considerations. Through careful and considerable effort, however, these limitations can be overcome.

# References


[1] SensAble Technologies Incorporated, http://www.sensable.com

[2] Raymond, E. S. & Young, B., The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O'Reilly & Associates, Cambridge, MA, January, 2001.

[3] NVIDIA Incorporated, http://www.nvidia.com


# Acknowledgements


The authors wish to thank SensAble Technologies, especially Ryan Toohil and Tom Ellery, for their support. The authors also wish to thank Tom Anderson of Novint for supporting this effort.

# Implementing Haptic Feedback in a Projection Screen Virtual Environment

**Andrew G. Fischer**
**Judy M. Vance**[1]
Department of Mechanical Engineering
Virtual Reality Applications Center
Iowa State University
Ames, Iowa 50011
fischea1@vrac.iastate.edu
jmvance@vrac.iastate.edu

## Abstract

Haptics refers to the sensing of force, weight, or other physical properties through feeling and touch. The additional information from haptic feedback makes certain engineering design tasks, such as part assembly, much easier than with a traditional computer interface. Haptic feedback, when used in a virtual reality application, is most often combined with either a stereo monitor or head mounted display. In this research, the PHANToM 1.5 haptic device is combined with a projection screen virtual environment, the C6 at Iowa State University, in order to explore the benefits haptic devices bring to this type of immersive environment. To achieve this goal several concepts are presented including: the design of a stand to support the PHANToM in the environment, a virtual volume to scale the PHANToM's physical workspace to a user defined portion of the virtual world, and an application to integrate the PHANToM's GHOST software with the vrJuggler virtual environment software. Two example uses of haptic feedback in the virtual environment are presented: a NURBS surface and a virtual assembly application. The assembly example uses Boeing's Voxmap PointShell (VPS) software to interface with the GHOST software and control the PHANToM. The benefits of haptic feedback in the virtual environment for these examples and some guidelines for using them are presented.

## Introduction

A key component of virtual reality (or VR) systems is the ability to immerse a participant in a computer generated virtual environment. Immersion refers to the sense of "being there" that a user feels in the virtual world; the greater the sense of immersion, the more real the virtual world appears [1]. The level of immersion a user feels in a VR environment is related to the number of senses stimulated, such as sight and hearing [2]. However, most virtual reality systems are lacking in a key area of stimulation, namely some form of physical or haptic feedback.

One device capable of providing haptic feedback in a virtual reality simulation is SensAble Technology's PHANToM [3]. This research examines the issues surrounding the use of the PHANToM in a six-sided projection screen synthetic environment, the C6 at Iowa State University.

## Implementation

Bringing the PHANToM into a projection screen virtual environment presents several challenges. Since the PHANToM is essentially a desktop device, using it in the large projection screen environment requires the PHANToM to be mobile and adjustable to accommodate a standing user. Second, the PHANToM's 19x27x37 cm physical workspace is much smaller than the projection screen environment's 10x10x10 ft size. Some method must be used to make the PHANToM useful over large portions of the virtual environment. Finally, a program must be written to integrate the operation of the PHANToM with the software controlling the virtual environment.

[1] Corresponding Author

*Phantom Stand*

Physically supporting the PHANToM in the virtual environment was accomplished by designing and building a stand to hold the PHANToM. This stand rolls about on four castor wheels, which may be locked to keep the stand from moving when the PHANToM is in use. Stand height is adjustable from 28 to 42 inches to accommodate different users and postures. Since knowing the orientation of the phantom stand is desirable, magnetic tracking devices, such as the Ascension Technologies MotionStar used in the C6, should be compatible with this stand [4]. Since magnetic materials adversely affect the accuracy of such trackers, the phantom stand was constructed out of bonded PVC plastic and stainless steel hardware. When the stand is in the virtual environment a magnetic tracking device is attached to one of the legs. A model of the stand appears in Figure 1.

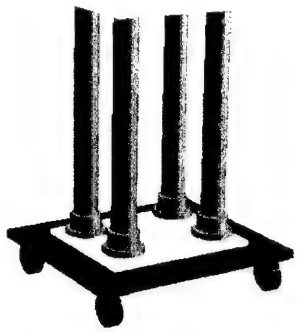


Figure 1. Phantom stand

*Phantom Volume*

To address the mismatch between the PHANToM's physical workspace and the size of the virtual environment, the concept of a phantom volume is presented. This consists of a user-defined quadrilateral volume of space in the virtual environment that correlates motion of the PHANToM's physical endpoint to a virtual position in the environment. This approach is similar to that taken by Preshce and Hirzinger in their work on workspace scaling for teleoperation [5].

A phantom volume is defined by selecting two opposite corners in the virtual space with a wand. The known orientation of the phantom stand, obtained from a magnetic tracker, is used to orient the volume. To aid selection, the volume is dynamically drawn between the first point and the current wand position until a second point it chosen. The completed volume may then be translated about the virtual world and/or scaled to a desired size. Figure 2 shows a phantom volume with a red sphere representing the virtual PHANToM endpoint.

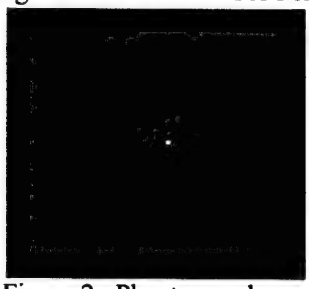


Figure 2. Phantom volume

When using the PHANToM, the virtual endpoint position is confined to the phantom volume, just as the physical endpoint is confined by the PHANToM's physical workspace. Motion of the actual PHANToM is scaled to match motion of the virtual position. This way the limited PHANToM physical working volume can be matched and used over an arbitrarily large space in the virtual environment.

*Phantom Driver*

The C6 is controlled with the vrJuggler software, a complete framework for virtual reality applications that may be used on a variety of virtual reality devices [6]. Integration of the PHANToM's GHOST software and vrJuggler is handled by the phantom driver program. The phantom driver is a C++ class that builds the haptic scene graph, loads the haptic geometry, positions the virtual PHANToM endpoint in the virtual environment, scales motion of the PHANToM to the phantom volume, and communicates with the rest of the simulation.

The phantom driver's first step is defining the workspace limits of the physical PHANToM device. Since the phantom volume isn't constrained to be a cube, the phantom driver creates the PHANToM's physical workspace to match the form of the phantom volume, so a single scale value suffices to match motion of the PHANToM's physical endpoint to the virtual world. This value, *world_haptic_scale*, is determined from the largest dimension of the phantom volume and

the *max_workspace_size* parameter as in equation 1.

$$world\_haptic\_scale = \frac{max\_workspace\_size}{PV[largest\_dimension]} \quad (1)$$

The *max_workspace_size* represents the largest dimension of the PHANToM's physical workspace. In some cases it is useful to confine the PHANToM endpoint to a box of modest size, ensuring that the user doesn't run out of device travel or collide the endpoint with the bulk of the physical PHANToM. At other times it may be desirable to make the physical workspace limits much larger, allowing the PHANToM free movement throughout its entire range of travel. For this work the *max_workspace_size* is set to 120.0 millimeters, which prevents the PHANToM endpoint from colliding with the physical device or the phantom stand.

The phantom driver's also must construct the haptic geometry required by the application. This geometry may consist of simple primitives and/or polygonal meshes. Once the phantom driver has the workspace set up and the geometry loaded, it uses GHOST to build the haptic scene graph. A diagram of this scene graph appears in Figure 3.

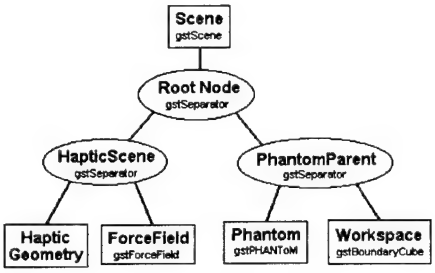



Figure 3. Haptic scene graph

A key feature of this scene graph is the placement of the Phantom and the Workspace objects into the PhantomParent separator. This allows the position of the PHANToM device and its workspace to be translated and rotated in the haptic space to match the position of the phantom volume simply by applying the proper transformations to the PhantomParent, since moving the PhantomParent separator moves both the PHANToM and it's Workspace.

**Example Applications**

In this research two example applications are presented that demonstrate the advantages of using the PHANToM in a projection screen virtual environment: a NURBS surface exploration example and a virtual assembly application that uses Boeing's VPS (Voxmap PointShell) software.

The NURBS surface example demonstrates using the GHOST software to build the haptic representation of an existing NURBS (Non-Uniform Rational B-Spline) surface. The geometry is then displayed in the virtual environment where the user may define a phantom volume around and interact with any portion of the surface. This application lets the user experience the additional information about an object's shape through the sense of touch while exploring the differences between the PHANToM's physical workspace size and the virtual phantom volume workspace.

The virtual assembly example has the user manipulate the PHANToM in an attempt to install a rudder pedal assembly into a simplified model for the lower front portion of a light aircraft. The haptic feedback provides a fast and intuitive way for a designer to determine if the assembly task can be completed. Boeing's VPS software is used to detect collisions between the pedal assembly and the aircraft structure. Physically-based modeling is used to calculate the haptic forces resulting from any collisions. GHOST is used to return these haptic forces to the PHANToM. The resulting program is capable of manipulating very complicated models at haptic feedback rates [7]. Figure 4 shows the virtual assembly example in use in the C6.

Figure 4. Virtual assembly application with PHANToM

## Conclusion

After using this application and experimenting with the examples, some conclusions can be drawn about how haptic feedback and the projection screen virtual environment aid the user in different tasks.

1. Haptic feedback makes manipulating a complex part through confined spaces faster and more intuitive than using a standard keyboard and mouse approach.
2. Being able to touch objects with the PHANToM provides additional information about the geometric structure, which may not immediately be noticeable visually.
3. The projection screen virtual environment makes interference issues encountered in a particular task easy to find and remedy.
4. Since several people may observe the virtual environment and share the PHANToM in the same simulation, the projection screen environment enhances collaboration between users.
5. For the examples presented in this work, the differences in workspace size between the physical PHANToM and the virtual phantom volume appear relatively unimportant.

While the addition of haptics to the virtual environment improved the ability of users to interact with digital models for the examples presented, there are some guidelines that should be followed to ensure a high quality haptic feedback experience in a virtual environment.

1. Avoid positioning much of the phantom in a location that does not align with the user's viewpoint.
2. Avoid making the phantom volume excessively large, as this reduces the quality of the force feedback.
3. Avoid positioning the phantom stand in the direct line of the user's sight.
4. Keep the PHANToM and its stand outside of the virtual geometry.


## Acknowledgements

This work was supported by NSF research grant DMII-9872604. The research work was performed using the facilities of the Virtual Reality Applications Center, Iowa State University.


## References


1. Pausch, R., D. Proffitt, and G. William. *Quantifying Immersion in Virtual Reality*. in *24th annual conference on Computer graphics and interactive techniques*. 1997: ACM Press.
2. Burdea, G.C., *Force and Touch Feedback for Virtual Reality*. 1996, New York: John Wiley & Sons, INC.
3. Massie, T. and J.K. Salisbury. *The PHANTOM haptic Interface: A Device for Probing Virtual Objects*. in *ASME Symposium on Haptic Interfaces for Virtual Environments*. 1994. Chicago, IL: ASME.
4. VRAC, *About the C6*. 2001, Iowa State University.
5. Preusche, C. and G. Hirzinger. *Scaling Issues for Teleoperation*. in *Fifth PHANToM Users Group Workshop*. 2000. Aspen, Colorado.
6. Bierbaum, A., et al. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. in *Virtual Reality, 2001*. 2001. Yokohama, Japan: IEEE.
7. McNeely, W.A., K.D. Puterbaugh, and J.J. Troy. *Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling*. in *26th Annual Conference on Computer Graphics and Interactive Techniques*. 1999.

# Haptic Interaction with 3D Ultrasound Data
# The e-Touch sono System

*Touch Your Baby Before He or She is Born*

**Walter A. Aviles**
**Thomas G. Anderson**
Novint Technologies
http://www.novint.com

## Abstract

The e-Touch sono™ system allows users to interactively feel and see 3D ultrasound images. This system is currently targeted for use in pre-natal imaging [1]. Parents are able to feel as well as see three-dimensional imagery. We have found that this increases both their understanding and their enjoyment of their child's ultrasound image. This has several important benefits. First of all, it allows physicians to more clearly explain the developmental process, progress and any complications. Second, it helps ease parental stress and anxiety over the progress of their child. Finally, it helps the all-important parent-child bonding process. Over time, the e-Touch sono system will be refined in order to be used for medical diagnosis, surgical planning, and intraoperative procedures.

## Background

Ultrasound is a high-quality, reliable and cost effective medical diagnostic tool. It has been used for medical imaging and diagnosis purposes since the Second World War [2][3]. The first medical ultrasound systems, mirroring their roots in Sonar technology, required that the patient be immersed in water for imaging to occur. They also used "A-mode" presentation of the ultrasound data -- blips on an oscilloscope screen. The more familiar "B-mode" presentation of two-dimensional gray scaled images followed shortly thereafter. Ultrasonic imaging came into more common use in the clinical environment in the 1960s and 1970s with the introduction of compact hand-held scanners with real-time B-mode imaging capabilities. More recently, medical ultrasound systems capable of generating three-dimensional (i.e., volumetric) data and images have become available.

Some ultrasound systems generate the three-dimensional images by analyzing and combining a series of two-dimensional B-mode images gathered using a hand-held scanner outfitted with a position/orientation tracker. Other systems utilize a more sophisticated transducer design to allow three-dimensional data to be gathered without requiring any external transducer movement or tracking. These later systems are often referred to as "4D" systems because they allow three-dimensional data and imagery to be gathered in real-time. GE Medical Systems' Voluson 730 ultrasonic imaging system, for example, currently generates 16 three-dimensional images per second [4].

One of the most common uses of ultrasound is in obstetrics and gynecology. Ultrasound has made it possible to non-invasively study pregnancy from beginning to end. Diagnosis of complications of multiple pregnancy, fetal abnormality and placenta previa became possible with the use of ultrasound in the clinical environment. Moreover, ultrasound exams are generally considered safe at the power levels used for diagnosis. In addition to providing physicians with a

useful diagnostic tool, ultrasonic imaging allows prospective parents to view their child's development.

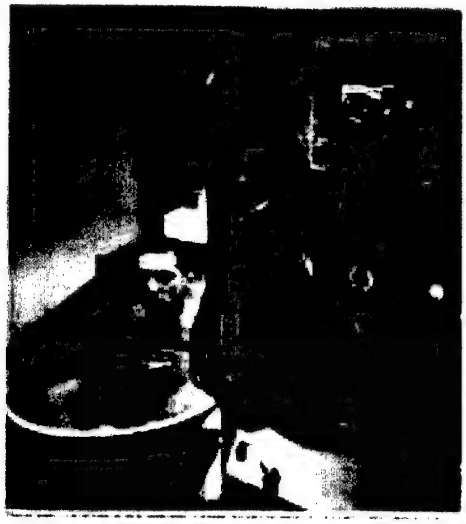



**Figure 1 -- Water Immersion Motorized B-mode Scanner Circa 1954**

## 3D Haptic Interaction

As part of our research and development, we have explored the use of haptic interaction in the understanding, use and modification of sensor-derived three-dimensional geoscientific [5] (e.g., seismic) and medical (e.g., CAT Scan or MRI) data. 3D ultrasound is particularly interesting because it is a safe and cost effective real-time imaging modality. Adding haptic interaction to 3D ultrasound promises to have many of the same benefits and features that we have seen before. Three-dimensional haptic interaction allows for the direct and unambiguous selection of parts of the ultrasound image for further exploration/analysis, simplified and direct six degree-of-freedom control of imaging tools such as interactive clipping planes and the use of an additional sensory channel or modality (i.e., touch) for understanding the data. These features hold the promise of simplifying and improving the clinical analysis of three-dimensional ultrasonic imagery by providing both a natural and a rich additional sensory interaction mode. Clearly, however, fully exploring and understanding the clinical impact and benefits of adding touch to ultrasound images will take time.

In order to help introduce haptic technology to the three-dimensional ultrasound community, we have developed a product, known as e-Touch(TM) sono, which is aimed at providing a more informative and enjoyable experience for parents when pre-natal 3D ultrasound images are taken of their child. Parents are able to feel as well as see three-dimensional imagery. We have found that this increases both their understanding and their enjoyment of their child's ultrasound image. This has several important benefits. First of all, it allows physicians to more clearly explain the developmental process, progress and any complications. Second, it helps ease parental stress and anxiety over the progress of their child. Finally, it helps the all-important parent-child bonding process.

This initial focus on parental interaction with ultrasound allows us to introduce haptic technology into the clinical environment in a way that has clear benefits today and allows the medical/diagnostic uses of haptic interaction with 3D ultrasound to be more gradually explored.

## The e-Touch sono™ System

The e-Touch sono System is a turnkey hardware and software system that allows users to interactively feel and see 3D ultrasonic images. Sono also allows the 3D images to be interactively cleaned-up and exported for the generation of 3D "hardcopy" or imagery.

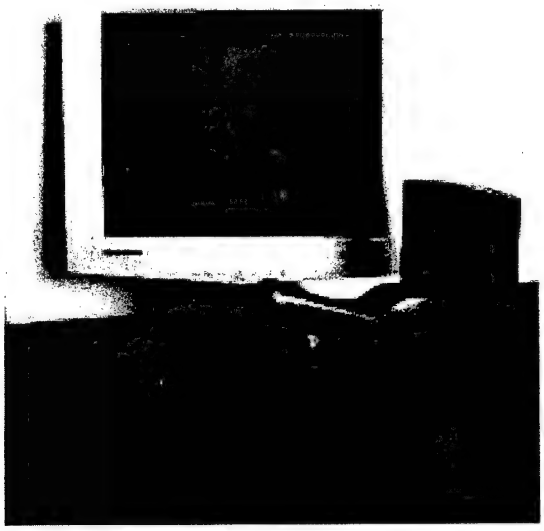


**Figure 2 -- The e-Touch sono System**

The system consists of a computer workstation, a graphical display, a 3D touch interface device and the e-Touch sono software. Sono can also be run on a laptop computer system.

**Figure 3 -- 3D Ultrasound Acquisition**

The overall viewing and haptic interaction process begins when a pregnant women gets a 3D ultrasound. The data, in volumetric form, is transferred to the e-Touch sono system. The

program automatically cleans up the data and applies default visualization and haptic parameters so that the "skin" surfaces in the data are visible and touchable. The parent or physician can then see and feel the surfaces of the child's face and body. Haptic skin textures are applied to the surfaces to improve the overall haptic experience. As shown in Figure 4, modern 3D ultrasound machines allow highly detailed images to be generated. Allowing parents to interactively control this imagery as well as "feel their baby" significantly adds to the overall educational, enjoyment and bonding aspects of the experience, in addition to easing anxiety about the health of the baby.

**Figure 4 -- Side by Side Images of Pre-Natal Ultrasound & Baby**

The key features of the sono system are:

- Reads data directly generated by GE Voluson 730 4D Ultrasound Systems
  - Cartesian Kretz V730 volume data file format via network or CDROM
  - Other data formats to be added
- Interactive visual display of 3D ultrasound data
  - Control color and opacity in real time
  - Real time rotate, translate & zoom
- Allows user to feel the ultrasound data in three dimensions
  - Surfaces (variable feel)
  - Volume properties (variable feel)
- Allows interactive clean-up of ultrasound data for 3D export in a variety of formats
  - Volume
  - Surface
  - Point
- Generates data for 3D "hardcopy" output
  - Several processes supported
- Generates data for 3D "take home" visualization
  - Several formats supported

## Technical Overview

### Data Pre-Processing

Three dimensional ultrasound data, like any real-world data, is noisy. Moreover, sampling resolution is such that voxel quantization is visible in the raw data obtained directly from the 3D ultrasound machine. As a first step in our visualization and haptic interaction process, we apply a standard set of three-dimensional filters to the ultrasound data. We utilize a 3D median filter followed by a gaussian smoothing process. The parameters of these filters were empirically

tuned for use with the Voluson 730 system but need not be changed across machines or images. Figure 5 shows a typical ultrasound image before and after filtering.

**Figure 5 – Typical 3D Ultrasound Before (Left) and After (Right) Pre-Processing**

**Visual and Haptic Rendering**

The sono system is a volume-based visualization and haptic interaction tool. Both modalities operate on exactly the same data set in real-time. Sono can generate surface representations, however, to allow for physical ("rapid prototyping") models to be produced.

***Visual Rendering.*** Visual volume rendering, allows internal structures to be viewed. Although this feature is not essential to the initial "parental" focus of the sono system, it is available for the exploration of the clinical diagnostic uses of the application. Volume visual rendering, being an image order technique, scales quite well with data set complexity. Modern consumer-level graphics cards provide solid interactive (i.e., greater than 10Hz) visual performance. As can be seen in Figure 5, the sono system's default visualization parameters are set so that low-density materials, such as amniotic fluid, are not visible.

***Haptic Rendering.*** Like visual rendering, haptic rendering is also performed directly using the volume data. Two forms of haptic interaction are currently provided -- a full-volume viscosity force feedback algorithm and an isosurface rendering algorithm. The viscosity force is meant as an early tool for physicians to explore the overall volume data set. The technique generates a viscosity-like force that varies with the haptic "density" applied to a particular data set value. The isosurface rendering algorithm allows analytic isosurfaces to be felt. It is the primary haptic rendering algorithm used for parental education and bonding purposes. Both algorithms operate based only on data in the local neighborhood of the haptic interaction point. This means that overall volume size or complexity does not affect their operation.

The isosurface algorithm consists of two parts. In the first part, the isosurface is found and surface compliance forces (i.e., normal to the isosurface) are generated. This algorithm can treat even "thin shells" as non-permeable. This is important for 3D ultrasound fetal images since data

for many parts of the image (e.g., face) is often in a thin shell form. Moreover, it is desirable to be able to vary the compliance model used for the surface independent of the whether or not the surface is part of a thin shell or deeper solid structure. For fetal imaging, the surface, for example, is rendered as quite soft or compliant using a simple spring model. The second part of the isosurface algorithm, applies surface texture (i.e., forces along the isosurface). We have tuned this model to provide a "skin-like" texture to surfaces.

***Surface Model Generation.*** The sono system can also generate surface representations of the isosurface. It currently generates stereolithography format (i.e., STL) files of the isosurface for either the entire data volume or for particular views of the data. This data can be used for manufacturing three-dimensional models of the baby for diagnostic and "keepsake" purposes.

## Conclusions

The e-Touch sono system allows three-dimensional ultrasound data to be felt as well as seen in real time. The initial focus of this application is to allow visual and haptic interaction with fetal images for parental education and bonding. This particular use of the system is quite compelling and has resulted in the commercial development and distribution of the sono system. Over time, we believe that, as the potential of haptic interaction for clinical diagnostic and analysis purposes is further explored, that sono will also have an important medical role.

## Acknowledgements

The authors wish to thank New Mexico Sonographics Inc. for allowing access to their ultrasound equipment in the early phases of the sono development. We also wish to thank the Novint technical team, especially Richard Aviles and Jake Jones, for their work on the sono effort and underlying technology.

## References

[1] Novint Technologies Incorporated, http://www.novint.com

[2] Woo, J, *A short History of the development of Ultrasound in Obstetrics and Gynecology*, http://www.ob-ultrasound.net/history.html

[3] *History of Ultrasound*, http://www.scmobileimaging.com/history.html

[4] GE Medical Systems, http://www.gemedicalsystems.com

[5] Aviles, WA and Ranta, JF, *Haptic Interaction with Geoscientific Data*, in Salisbury, JK and Srinivasan, MA (Eds), "Proceedings of the Fourth PHANTOM Users Group Workshop," AI Lab Technical Report No. 1675 and RLE Technical Report No. 633, MIT, November 1999, pp. 78-81.

# A 3-D Lasso Tool for Editing 3-D Objects: Implemented Using a Haptics Device

**Marjorie Darrah**
**Institute for Scientific Research**
**Fairmont, WV 26554**
**mdarrah@isr.us**

**Alicia Kime**
**School of Science and Mathematics**
**Fairmont State College**
**Fairmont, WV 26554**
**akime@mail.fscwv.edu**

**Frances Van Scoy**
**Lane Department of Computer Science and Electrical Engineering**
**West Virginia University**
**Morgantown, WV 26505**
**fvanscoy@wvu.edu**

## Abstract

Many paint programs include a lasso editing tool that allows the user to capture an irregular portion of a two-dimensional figure. However, no corresponding tool is known to exist for three-dimensional editing. This paper describes a lasso tool, based on a convex hull algorithm, that has been simulated using the SensAble™ PHANTOM™ haptic device. The PHANTOM™ device is employed as a three-dimensional mouse to select a set of non-planar voxels, that is, three-dimensional pixels. The algorithm uses the selected voxels to define a convex hull. Once the voxels within the convex hull have been identified, they can be deleted, copied, or modified.

## 1 Introduction

This project addressed the problem of selecting a set of voxels (three-dimensional pixels) in a three-dimensional model for editing. The tools available for two-dimensional editing are rather flexible; many paint programs allow the mouse to sketch the boundary of a region, either freehand or using line segments to form a polygon. The user does not have to enclose the region in a circle, ellipse or square but may select a shape that is irregular. We sought to generalize the two dimensional idea of a lasso enclosing a irregular polygon to a lasso for three-dimensional sets of voxels.

Current packages on the market allow three-dimensional data to be edited in several ways including what are known as: the cookie-cutter (or extrusion) approach, the melon baller approach and the two-dimensional slicing approach [1]. The cookie-cutter approach allows an arbitrary shape to be drawn on a plane to enclose an area, then extends the shape downward orthogonal to the plane so that it selects a volume much like a cookie cutter presses through a large quantity of dough. The melon baller method uses a regular shape, such as sphere, cube, or rectangular prism to carve out sections of the three-dimensional space by using that shape. This allows greater control over the volume selected than the previous method, but it still yields a regular shaped region. In a third approach, two-dimensional slicing, a two-dimensional projection of the three-dimensional data set is edited repeatedly. The data set may be rotated to observe the data from different viewpoints, but the mouse is still editing in two dimensions.

The tool we created and simulated using the PHANTOM™ haptics device makes possible the freehand editing of a three-dimensional data set. Our prototype allows the user to specify a data set, containing points with (x,y,z) coordinates, which is then rendered as voxels in a three-dimensional haptic scene. To construct a convex hull, the haptics device is first used to select a set of three-dimensional voxels. When at least four non-planar voxels have been chosen, the program will construct a convex hull that encloses all points within the boundary of an irregular convex polyhedron defined by those voxels.

## 2 Mathematical Approach

To create a lasso tool that allows for the arbitrary selection of points and the enclosure of irregular three-dimensional volumes, we first explored the geometry of the problem. Just as the two-dimensional lasso tool uses a polygon to enclose the area to be edited, a three-dimensional lasso must define a polyhedron. A polyhedron is a region of space whose boundary is composed of flat polygon faces, any pair of which are either disjoint or meeting at edges and vertices.

It should be noted that we here are discussing and constructing a *convex* polyhedron. A set **S** of points is defined as convex if $\mathbf{x} \in \mathbf{S}$ and $\mathbf{y} \in \mathbf{S}$ implies that the line segment $\mathbf{xy} \subseteq \mathbf{S}$. A polyhedron that is constructed from the user-selected points is called a *convex hull* of the set. The following example may help visualize the concept of a convex hull. In a two-dimensional plane, the convex hull is the shape a string assumes when anchored to a nail at the lowest point with respect to y and wrapped

counterclockwise around nails pounded into the plane at each point. In three dimensions, the *boundary* of a convex hull is the shape taken by plastic wrap stretched tightly around all the points. More formally a definition of convex hull of a set **S** is the intersection of all convex sets that contain **S**.

Many algorithms exist for constructing both two and three-dimensional convex hulls. Graham's Algorithm [1], the fastest method for constructing a convex hull in two dimensions, has no obvious generalization to three dimensions. Among the best-known three-dimensional algorithms are Gift Wrapping (Chand & Kapur), Divide and Conquer (Preparata & Hong) and Incremental Algorithm (Seidel & Kallay). We chose to implement the Incremental Algorithm outlined by Joseph O'Rourke in the second edition of his, *Computational Geometry in C* [2].

In two dimensions the basic method of the Incremental Algorithm is to add points one at a time, constructing the hull of the first $k$ points at each step and using that hull to incorporate the next point.

Algorithm: INCREMENTAL ALGORITHM

Let $H_2 \leftarrow \text{conv}\{p_0,p_1,p_2\}$.

For $k = 3$ to $n$-1 do

    $H_k \leftarrow \text{conv}\{H_{k-1} \cup p_k\}$

The algorithm begins with a triangle and considers whether the next point is inside or outside the area enclosed by that triangle. If the point is inside, then it is discarded; if it is outside, then the algorithm determines the two tangent lines from the new point to the triangle. The algorithm continues in this manner until the convex hull is fully defined.

This incremental algorithm, with time complexity of $O(n^2)$, can be generalized to three dimensions.

Algorithm: 3D INCREMENTAL ALGORITHM

Initialize $H_3$ to tetrahedron $(p_0,p_1,p_2,p_3)$.

for i = 4, ..., $n$-1 do

for each face f of $H_{i-1}$ do

    Compute volume of tetrahedron determined by $f$ and $p_i$.

    Mark $f$ visible iff volume < 0.

    if no faces are visible

        then

            Discard $p_i$ (it is inside $H_{i-1}$).

        else

            for each visible border edge $e$ of $H_{i-1}$ do

                Construct cone face determined by $e$ and $p_i$.

            for each visible face $f$ do

                Delete $f$.

            Update $H_i$.

The overall structure of the three-dimensional incremental algorithm is identical to that of the two-dimensional version. From a set of identified points, the algorithm chooses four non-planar points from which the initial hull is constructed. At the *ith* iteration, the algorithm computes the hull-in-progress by adding a new point. This action yields two possibilities: (1) if the new point is inside the existing hull, it is discarded; (2) if it is outside, the algorithm constructs a cone face determined by the new point and each border edge visible from that point. The process of defining the convex hull on the basis of a single new point $p$ is visualized in Figures 1 - 3.

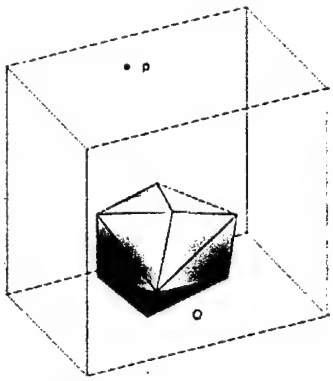



Figure 1: Convex hull $Q$ and point $p$ outside the hull

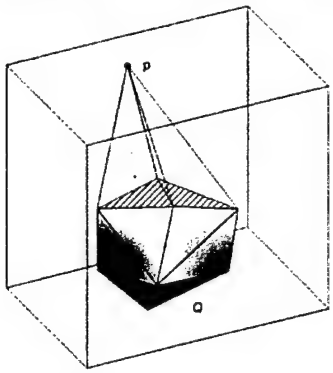



Figure 2: The striped faces are interior and marked for deletion when the hull is extended to include point $p$.

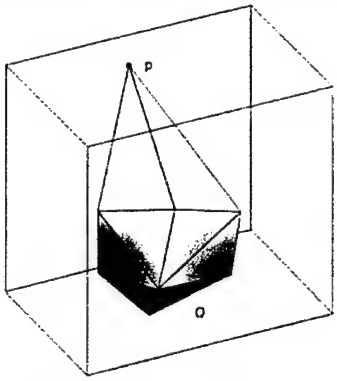



Figure 3: The hull after point p and the new faces have been added.

## -3 Implementation of Convex Hull Algorithm

To simplify the data structures used, the algorithm assumes that every face of the surface of the *polytope* (convex polyhedron) is a triangle. Three primary data types are required to define a convex hull, vertices, edges, and faces. The computer program that implements the algorithm consists of four basic sections: read, create initial polytope, construct the hull, and print. First the vertices are read into an array. Next, an initial polytope for the incremental algorithm is created. It is a double-sided triangle, a polyhedron with three vertices and two faces that are identical except for the order of their vertices. To construct this figure, three noncollinear points are found, and after marking them as processed the double-sided triangle is built as the first hull $Q$. It is now necessary to find a fourth nonplanar point $p$ to form a tetrahedron. For each point $p$ it must be determined if $p$ is inside or outside $Q$. To accomplish this, one must determine for every face $f$ of $Q$ if the face $f$ is *visible* from point $p$. The face $f$ is visible from $p$ iff it lies in the positive halfspace determined by the plane containing $f$. The positive side is determined by the counterclockwise orientation of $f$. If no face is visible from $p$, then it must lie inside the hull, and it is marked for deletion. If $p$ is outside $Q$, then the hull is transformed by finding the tangent planes that bound a cone of triangle faces with the apex at point $p$ and the base edges of $Q$ (Figure 2). The portion of the polytope visible from $p$ forms a connected region of the surface of the existing hull as indicated by the striped faces in Figure 2. The interior faces of the region must now be deleted and the cone connected to the boundary. To determine the edges of the hull, those edges adjacent to two visible faces are marked interior and marked for deletion; those edges with one adjacent visible face are identified as on the border of the visible region. Once the edges of the border are identified, then for every edge on the border a new triangular face can be constructed using that edge and the point $p$. At this point, the convex hull is almost complete. All that remains is to "clean up" the hull by deleting hidden faces and edges and making sure the vertex, edge and face lists are properly linked.

## 4 Construction of the Prototype

Our effort to model the algorithm began by building a visualization program using OpenGL. The program we devised first reads in a data set consisting of (x,y,z,r,g,b) giving position and color information and then creates a voxel for each point in an $n$ x $n$ x $n$ space. The voxels are cubes that can be opaque or transparent. Once the scene has been rendered, it can be rotated clockwise or counterclockwise in the x, y, and z directions around a fixed point. Given a set of three-dimensional non-planar coordinates (a subset of the original data set) the program determines the convex hull defined by those points. In our program we identified the hull by changing the color of the points that constituted it. When operational the program demonstrated that the convex hull algorithm could be implemented in a graphics program.

Wishing additionally to implement the model into a three-dimensional touch environment, we developed a related program using the GHOST® API to incorporate haptics. The program we developed for use on the PHANTOM™ haptics device uses "voxel world" coordinates, where the position of the voxel is (x, y, z) and x, y, and z are all integers. As a trial experiment, we created a 3 x 3 x 3 grid of spheres to represent the voxel space. The spheres were large enough to be felt, but separated by sufficient space to permit movement of the haptic cursor within the grid. We made use of the PHANTOM™ haptics device as a three-dimensional selection device. Using the stylus, we selected certain spheres that represented the set of points to define our convex hull. The algorithm then determined the interior points of the hull and we identified those voxels by changing their color.

## 5 Future Work

One application of a tool such as we devised is to facilitate the exploration of three-dimensional scientific or abstract data sets using haptics. At the present we hope to incorporate this work into a tool kit that uses haptic feedback to explore Light Detection and Ranging (LIDAR) data. The LIDAR tool kit will enable the user to be immersed in and interact with a point cloud of data and the 3-D lasso editing tool will allow for selection of points for removal or coloring.



## References

[1] Van Scoy, F., Peredera, A., and Kime, A. (1999), "A 3-D Lasso Tool for Editing 3-D Objects: Preliminary Work", Workshop on Virtual Reality, Marilia, Brazil, November 18-20.

[2] O'Rourke, J. (1998), *Computational Geometry in C*, second edition, Cambridge University Press, Cambridge.

[3] Graham, R.L. (1972), "An efficient algorithm for determining the convex hull of a finite planar set", *Inform. Process. Lett.* **1**, 132-3.

# Touch&Tell: A game-based tool for learning to use the PHANToM

Eric Acosta, Bharti Temkin PhD
*Department. of Computer Science, Texas Tech University*
*Lubbock, TX 79409*
*Bharti.Temkin@coe.ttu.edu*

**Abstract**

We have developed a game, Touch&Tell, with the objective of teaching how to use the PHANToM. The game starts with simple 3D shapes and leads to progressively more complex 3D objects, including anatomical structures. The purpose is to overcome some of the problems encountered by inexperienced users, such as excessively large hand movements and loss of the object contact, and to increase the user's sensitivity to computer generated sense of touch. Visual and audio cues are initially included in order to help locate and stay in contact with the object; the cues can be removed as the skill improves.

The Touch&Tell game can be also used to collect data that elucidates decomposition of the sense of touch into sensory components. To this purpose the 3D object can be made invisible in order to help identify the effects of the visual component of haptics. Haptic skills can be monitored and recorded and progress evaluated for different levels of complexity. This makes it also possible to quantify the effectiveness of point-based haptic devices.

## 1. Introduction

Haptic sense, based on distributed and kinesthetic receptors, provides information about objects and surfaces being touched [1,2]. The sense of touch is a complex phenomenon involving many senses and the perceived sense of touch is subjective and dependent on physiological and psychological factors. For example, human psychophysical experiments have shown that both haptic and visual systems can create biases in virtual environments when used alone, but compensate for each other when used jointly [3]. Visual information can alter the haptic perception of spatial properties such as size, location, and shape [4]. Humans rely more on the visual than kinesthetic cues when the visual information conflicts with the haptic information for both spatial and stiffness information [3,5].

In real life, familiar objects can usually be identified haptically (without vision) with virtually no error within a period of 1-2 seconds [1]. Object exploration tends to begin with general-purpose procedures that provide coarse information about the object and lead to specialized procedures to look for distinctive features of the unknown object [1]. Simplifying the haptic stimuli to a point requires more exploration time to acquire a complex object's feature information.

Given the complexity of the human haptic sensory system, simulating touch in a virtual environment is likely to be difficult. During haptic application demonstrations, new users struggle to learn the use of haptic device and consequently have harder time performing specific application tasks. Touch&Tell was developed to teach the use of the device first, thus preparing the user for more complex application tasks.

## 2. Touch&Tell

Figure 1. Touch & Tell game.

Initially, the user is shown how to move the haptic device to touch some simple 3D objects. After gaining some experience, the user is allowed to explore an object that is not displayed graphically. Separating the haptic and visual component requires the user to rely on the sense of touch to identify the object in question.

The device is used to touch an extendable library of objects, which is broken down into three groups of increasing complexity. These groups consist of solid geometric shapes, engraved alphabet blocks, and anatomical structures, Figure 1. As users explore the hidden object, they look for key characteristics that can be used for identification. For example, a cylinder would be identified if the object had a curved side and a flat top and bottom. The identification process forces controlled device movements in order to search for object's characteristics and thus refines user's control of the device. The geometric shapes help in general discrimination, alphabet objects require fine discrimination, and anatomical structures require both general and fine discrimination. In order to be successful, the user has to develop a strategy for object recognition.

An image key, shown in Figure 1, is provided to the user as a visual reference to aid in identifying the hidden object. After identifying the object, the user can press the corresponding haptic button on the image key to identify the object. A correct answer causes a bell sound, displays the hidden object, and then brings up the next hidden object. An incorrect decision causes a buzzer to sound and allows the user to make another guess.

### 2.1 Visual and audio cues

Optional visual and audio cues are included to help locate and stay in contact with the hidden object. The first visual cue is a line directing the proxy's graphical representation towards the center of the object. The second visual cue is a directional indication that tells the user whether the cursor is in front, behind, above, below, left, and/or right of the object. The combinations of the cues work well to reduce the level of difficulty for locating the object.

Once contact with an object is made the cursor turns green and a sound is played. When contact is lost, the cursor returns to the initial white color and another sound is played. These cues help the user to stay in contact with the object.

### 2.2 Changing the level of difficulty

An experienced user can choose to manipulate the objects (e.g. rotate, translate, and scale) and work at a more difficult level without the cues. There are other variable modes such as where the object category is known/unknown and the object is hard/soft. Using combinations of the different modes allows for increasing/decreasing the difficulty level of the task. For example, knowing the object's category reduces the number of possible choices and decreasing the stiffness of the object causes surface details to become less apparent. Other modes can be incorporated as needed for different experiments.

### 2.3 Data collection

In order to keep track of who is playing the game, the user must log in. All of the user's guesses are logged with a timestamp to determine the number of errors that occurred and the amount of time taken while identifying the unknown object. Two different timings are recorded. The first time is how long it takes to locate the object and the second is how long it takes to identify the object after the initial contact. Separation of these two times allows for a more accurate reading of how long the user actually explores the object in order to identify it. Touch&Tell also records contact information to keep track of the number of times the user gained/lost contact with the object. This helps in determining the ease of following the surface of the 3D object. Changes in strategy and user's difficulty level are also recorded. Based on a set of weighted parameters, the user's achieved level is calculated and stored.

## 3. Discussion

Although Touch&Tell is discussed in this paper as a tool for instructing users how to use the Phantom, it has many more potential applications. Touch&Tell can be a haptic research tool. Through various experiments, researchers trying to understand what roles the physiological and psychological factors (including vision) play for the sense of touch would benefit from this game. Other experiments, such as one performed by other researchers [6,7] to measure the Just Noticeable difference (JND) for force feedback, can be carried out. Researchers interested in the finer surface features [8] may be able to run experiments and see how effective the fine surface features are for identifying the objects. Experiments with Touch&Tell are expected to quantify how effective the sense of touch through a point-based device is for identifying 3D objects. This type of study will be beneficial for applications for visually impaired users.

## 4. Conclusions

Exploring a 3D environment with a point-based haptic device causes problems for new users. The Touch&Tell game teaches how to use the Phantom and how to control device movements in order to identify hidden objects by touch. Users utilize both general and fine discrimination in order to locate key characteristics of hidden objects for identification. This fun and challenging game is working as an effective tool to introduce haptics to new users.

Touch&Tell's data collection capabilities also make it possible to use it as a haptic research tool. Depending on the specific aim, Touch&Tell can be tailored to record different types of information.

A version of Touch&Tell called "ifeelit" is displayed as an exhibit in the Future Technology Center that Telecom Italia has created in Venice Italy.

## 5. Acknowledgements

The authors would like to thank Dr. Parvati Dev of the SUMMIT group at Stanford University and Dr. Mike Ackerman of NLM for their support; especially Dr. Leroy Heinrichs for being very enthusiastic and providing many great suggestions. This work was supported in part by the Surgery Department of the Texas Tech Health Sciences Center and the State of Texas Advanced Research Project Grant 003644-0117.

## 6. References

1. Klatzky, R., "Haptic Perception", MIT Cognet, http://cognet.mit.edu/MITECS/Entry/klatzky.html.
2. Burdea, G. C., *Force and Touch Feedback for Virtual Reality*, New York: Jon Wiley & Sons, INC., 1996.
3. Wu, W-C., Basdogan , C., Srinivasan, M. A., "Visual, Haptic, and Bimodal Perception of Size and Stiffness in Virtual Environments", *ASME Dynamic Systems and Control Division*, (DSC-Vol.67): 19-26, 1999.
4. Heller, M. A., Schiff, W., "The psychology of Touch", 1991.
5. Durfee, W. K., Hendrix C. M., Cheng, P., Varughese, G., "Influence of Haptic and Visual Displays on the Estimation of Virtual Environment Stiffness", *ASME Dynamic Systems and Control Division*, (DSC-Vol. 61): 139-144, 1997.
6. Srinivasan, M. A., Chen, J-s., "Human Performance in Controlling Normal Forces of Contact with Rigid Objects", *ASME Advances in Robotics, Mechatronics, and Haptic Interfaces*, (DSC-Vol. 49): 119-125, 1993.
7. Allin, S., Matsuoka, Y., Klatzky, R., "Measuring Just Noticeable Differences for Haptic Force Feedback: Implications for Rehabilitaion", *Proceedings of the 10$^{th}$ Symp. On Haptic Interfaces For Virtual Envir. & Teleoperator Systs. (Haptics '02)*, March 24-25, 2002.
8. West, A. M., Cutkosky, M. R., "Detection of Real and Virtual Fine Surface Features With a Haptic Interface and Stylus", *ASME Dynamic Systems and Control Division*, (DSC-Vol. 61): 159-166, 1997.

# Assessing the increase in haptic load when using a dual PHANToM setup

Joan De Boeck, Chris Raymaekers, Karin Coninx

Expertise Centre for Digital Media, Limburg University Centre,
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{joan.deboeck, chris.raymaekers, karin.coninx}@luc.ac.be

## Abstract

*Two handed input and collaboration are currently two active research areas in the domain of virtual environments. In a haptic application in particular, by means of a PHANToM device, two possibilities to build such a setup exist: either by integrating two PHANToM interface cards in one computer, or by using two networked computers, each equipped with one PHANToM interface card. The former method is likely to increase the computer's haptic load, while the latter adds extra code complexity and latency in the interaction, but does not augment the haptic load. Since very little can be found about the consequences of both solutions, this paper will describe an experimen, that measures the increase in haptic load of a dual PHANToM setup over a standard configuration.*

## 1 Introduction and related work

Although the use of force-feedback is a huge improvement in the interaction with virtual environments, current implementations are mostly restricted to a single point in space. To achieve a second interaction point, alternatively, a second interface card can be installed in order to drive a second PHANToM device, but this is very likely to increase the computer's haptic load. On the other hand, when using a collaborative application, multiple computers, distributed across the network, are all connected to their own haptic device [ALHAL01][HESPA00]. However, this setup adds extra code complexity and latency in the interaction. As an example, our research into the virtual percussionist application [DEBOE02] adopts those principles of distributed collaborative setups and applies them in a two handed input application. We believe this extra coding complexity might be justified when the increase in the haptic load gets too high in the single computer solution. At this moment however, no formal research has been conducted into this increase in a dual PHANToM setup. Some experiments on the computational load in a single PHANToM setup have been performed: Acosta et al [ACOST02] measured the maximum complexity in terms of number of objects and object complexity across different GHOST versions. Anderson et al [ANDER02] compared the performance of the GHOST API with e-Touch API for large complex objects. This paper, as a part of our research in two-handed haptic input, extents the above-mentioned research. The next sections will describe an experiment, which compares the haptic load of a dual PHANToM setup and a single PHANToM setup and discusses on the results.

## 2 Experimental Setup

This paper elaborates on two experiments. In a first experiment, we have measured the haptic load in a scene that contains one single object with increasing complexity. This object has been created by subdividing either a tetrahedron or a cube using the Loop subdivision scheme [LOOP87][RAYMA01]. Each subdivision level is represented in the haptic scene graph, by an instance of gstTriPolymeshhaptic. In order to test if the haptic load depends on the object's geometry, we also have used more natural models like a rabbit and a fish.
In a second test, we measured the haptic load in a scene with an increasing number of objects. These objects are positioned in a 3D matrix (as in [ACOST02]) which can grow in each direction. The objects do not intersect each other's bounding box and they also varied in complexity, using the same techniques as in the first experiment.
Both experiments have been conducted with a single and a dual PHANToM setup. The criterium measured in the two experiments is the haptic load, as measured with Sensable's Haptic Load (HLOAD) tool.

The computer used in our experiments was a Pentium III 600 MHz, 256 MB RAM, running Windows NT SP6. However, due to an incompatibility between the PHANToM PCI interface card, the AGP adapter and the computer's motherboard, we were forced to conduct our tests with a poor video card. Since the GHOST thread is a high priority thread, we believe this has little or no effect to the haptic load measured.
Our tests have been conducted on Windows NT, because we did not succeed in connecting two PHANToM devices on Windows 2000[1]. Our assumption that the choice of our hardware and OS does not influence our results are confirmed by comparing our single PHANToM test results on a Pentium III 850 MHz and a dual Pentium III 800 MHz running Windows 2000.

## 3 Results

### 3.1 Experiment 1: Objects with increasing complexity

| | | | 1 Phantom | | | 2 Phantoms | | |
|---|---|---|---|---|---|---|---|---|
| Subdivision | | #tri | N Contact *1P/0C* | Contact *1P/1C* | Contact & Moving | N contact *2P/0C* | 1 Contact *2P/1C* | 2 Contacts *2P/2C* |
| Tetraedron | level 0 | 4 | <10 | >10 | | 10 | 10< - <20 | <20 |
| Cube | level 0 | 12 | <10 | >10 | | >10 | 10< - <20 | <20 |
| Tetraedron | level 1 | 16 | <10 | >10 | | >10 | 10< - <20 | <20 |
| Cube | level 1 | 48 | <10 | >10 | | >10 | 10< - <20 | <20 |
| Tetraedron | level 2 | 64 | <10 | >10 | | >10 | 10< - <20 | <20 |
| Cube | level 2 | 192 | <10 | >10 | | >10 | 10< - <20 | 20 |
| Tetraedron | level 3 | 256 | <10 | >10 | | >10 | 10< - <20 | 20 |
| Cube | level 3 | 768 | <10 | >10 | | >10 | 10< - <20 | >20 |
| Tetraedron | level 4 | 1024 | <10 | >10 | | >10 | 10< - <20 | >20 |
| Cube | level 4 | 3072 | <10 | >10 | | >10 | 10< - <20 | >20 |
| Tetraedron | level 5 | 4096 | <10 | >10 | | >10 | 10< - <20 | >20 |
| Cube | level 5 | 12288 | <10 | >10 | | >10 | >20 | >30 |
| Tetraedron | level 6 | 16384 | <10 | <20 | | >10 | >20 | >30 |
| Cube | level 6 | 49152 | <10 | 20 | 30 | >20 | 40 | <60 |
| Tetraedron | level 7 | 65536 | <10 | 30 | 40 | <20 | 40 | >70 |
| bunny | | 69451 | <10 | 50 | 60 | <20 | 70 | quit |
| Fish | | 100480 | <10 | 30 | 90 | 20 | Unstable | quit |
| rabbit | | 134074 | <10 | 80 | quit | <20 | quit | quit |
| Thetraedron | level8 | 262144 | <10 | 70 | quit | quit | quit | quit |

Table 1: haptic load in a single and double PHANToM Setup with complex objects.

Table 1 summarizes the results of the first experiment. The values indicate a percentage of the haptic load of the simulation. When looking at the first column where no contact is made, one can see a quite constant haptic load both in the single PHANToM (1P/0C) as in the dual PHANToM setup (2P/0C). When touching the object (without moving the surface contact point over the surface) (1P/1C) the haptic load starts increasing from a shape with 160,000 triangles. This is consistent with the values reported in [ANDER01]. As can be seen in the next column, the haptic load augments when the surface contact point moves over the object's surface. This causes the GHOST-thread to quit with the most complex models in our experiment.
With the dual PHANToM condition, the haptic load again is quite stable when no contact is available (2P/0C), but increases slightly with increasing the object's complexity. The haptic load when one PHANToM touches the object (2P/1C) is somewhat higher, compared with the single PHANoM setup. The table here indicates a higher increase for the more complex models. The second surface contact points introduces another increase in such that the total haptic load is roughly 50% more than in (1P/1C). The results of those experiments are graphically depicted in fig 1.

[1] Another research lab reported us the same technical problems when running Windows 2000.

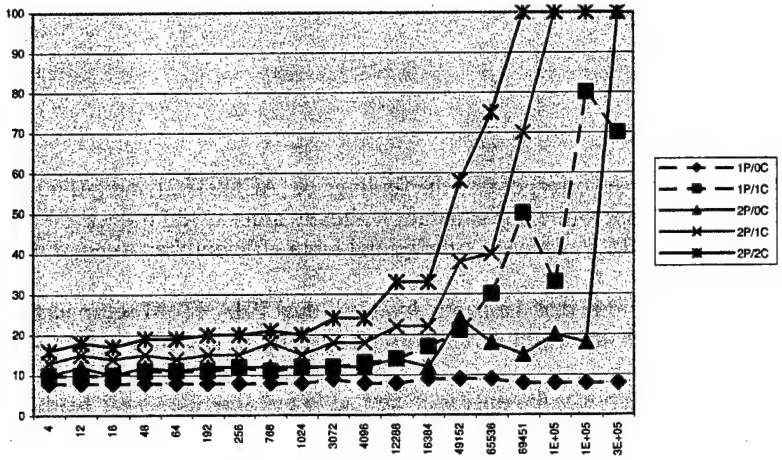



Fig. 1. Graph of the haptic load in a single and double PHANToM Setup with complex objects.

### 3.2 Experiment 2: Scenes with increasing complexity

| Tetrahedron Subdivision Level 0 (4 triangles) | | | | | |
|---|---|---|---|---|---|
| | One PHANToM | | Two PHANToMs | | |
| **# objects** | **No Contact** | **1 Contact** | **No Contact** | **1 Contact** | **2 Contacts** |
| 8 | 10 | <20 | >20 | >30 | >40 |
| 27 | <20 | >30 | 30 | <50 | >60 |
| 64 | >30 | <70 | <60 | quit | quit |
| 125 | <50 | quit | quit | quit | quit |
| 216 | quit | quit | quit | quit | quit |

Table 2: haptic load with multiple objects in the scene. (subdivision level 0)

| Tetrahedron Subdivision Level 2 (64 triangles) | | | | | |
|---|---|---|---|---|---|
| | One PHANToM | | Two PHANToMs | | |
| **# objects** | **No Contact** | **1 Contact** | **No Contact** | **1 Contact** | **2 Contacts** |
| 8 | >10 | 20 | <20 | >20 | 30< - <40 |
| 27 | <20 | 30< - <40 | >30 | >50 | <70 |
| 64 | >30 | 50< - <60 | <70 | quit | quit |
| 125 | <70 | quit | quit | quit | quit |
| 216 | quit | quit | quit | quit | quit |

Table 3: haptic load with multiple objects in the scene. (subdivision level 2)

| Tetrahedron Subdivision Level 4 (1024 triangles) | | | | | |
|---|---|---|---|---|---|
| | One PHANToM | | Two PHANToMs | | |
| **# objects** | **No Contact** | **1 Contact** | **No Contact** | **1 Contact** | **2 Contacts** |
| 8 | 10 | <20 | 70 | quit | quit |
| 27 | 20 | >40 | quit | quit | quit |
| 64 | <40 | 60 | quit | quit | quit |
| 125 | <70 | quit | quit | quit | quit |
| 216 | quit | quit | quit | quit | quit |

Table 4: haptic load with multiple objects in the scene. (subdivision level 4)

Tables 2, 3 and 4 show the results of the haptic load in a scene with an increasing number of objects. Each table displays the same number of objects, but with more triangles per object. All the objects are placed in a grid in such a manner that their bounding boxes do not overlap. A single PHANToM setup can fully support a scene with up to 64 tetrahedrons, while the dual PHANToM setup has the same load when simulating only 27 objects. The results of table 4 show that the haptic loop with 2 PHANToMs quits when touching one of the level 4 subdivision-objects in the scene, while the single PHANToM setup still supports 64 objects.

## 4 Discussion

### 4.1 Results

Our values in the single PHANToM case correspond to the findings of [ACOST02] and [ANDERS02]. Although, [ACOST02] can support up to 600 cubes, while our simulation already quits at 125 objects. We suppose this is caused by the standard *gstCube* (used in [ACOST02]), which is simpler and more efficient than the *gstTriPolymesh* used in our experiment.
From the first experiment, we can conclude that the haptic load in a dual PHANToM setup has been increased, compared to the single PHANToM setup. As long as the object is relatively simple (up to 16,000 triangles) the haptic load keeps below 30%, which results in a stable simulation.
Although the HLOAD application, which was the only tool we had to measure the haptic load, is not the most accurate tool one can imagine, we roughly can say that the second PHANToM increases the haptic load with about 50%. This can be confirmed by comparing the number of triangles in the most complex, but stable simulation in the first condition (fish with 100,480 triangles) with the maximum number of triangles in a stable dual PHANToM simulation (tetrahedron level 7 with 65,536 triangles).
Even more pronounced is the increase of the haptic load in a scene with multiple objects. If we compare "single contact with one PHANToM" (1P/1C) with a "double contact with two PHANToMs" (2P/2C), we see that the haptic load almost doubles. This makes that complex scenes, which can be run with one PHANToM, are not supported by a dual PHANToM setup.

### 4.2 Other findings

During the course of our experiments, we have encountered a number of interesting situations. Some of these appear to be obvious, but we believe they can give the programmer a better understanding of optimizing a more complex scene.

- When starting the haptic loop, very often a "haptic load spike" is encountered. This is why heavy models often crash at the beginning of the simulation. However if a complex simulation accidentally "survives" a startup, the simulation seldom is completely stable. Most of the time the haptic thread quits when acting on the scene.
- When exploring our "natural" objects (fish, rabbit), the haptic load was higher when approaching a more complex region with lots of small triangles.
- When sliding the PHANToM over an object, this increases the haptic load, compared to a static contact. On the other hand a static contact, which touches more than one triangle (e.g. in a corner), requires more processing time.
- We could not make Windows 2000 to work with the two PHANToMs, although we have conducted some of our single PHANToM tests on two Windows 2000 PCs, as well. In general we can state that there is little difference between the results of these tests and the Windows NT test.
- At first sight, the dual processor seems to perform better in starting-up a very complex scene: scenes that quit at start-up, do run on the dual processor computer. When interacting in those scenes, however, the results are quite similar to a single processor machine.
- On a dual processor computer other tasks will slow down less when executing a heavy GHOST thread. For instance, when running complex scenes, the haptic loop will slow down the graphics on a single processor computer, which is not true on a dual processor machine. This is quite obvious because the ghost thread in some cases can take up to about 99% of one processor's time, which is only 50% of the total processing power of a dual machine.

## 5 Conclusions and future work

For a multiple-contact interaction with a PHANToM device, two possibilities exist: most commonly, a second PHANToM will be attached to the same computer, or alternatively two computers in a network, both connected to a separate PHANToM share the same virtual scene. The first solution increases the computational load; the next solution introduces network delays. As a step to a well-grounded choice between those two options, in this paper we have conducted a test to measure the increase of the haptic load of a common dual PHANToM setup. Because we could not make a dual setup to work under Windows 2000, we have conducted our experiment under Windows NT SP6.

The experiment consisted of two tests: one that measured the haptic load in respect to the complexity of one object, and a second experiment, which tested the haptic load in respect to the scene complexity.
We can conclude that the increase of the haptic load is quite significant (roughly about 50% in experiment 1 and about 100% in experiment 2), but this is of less importance when running small scenes (1 object of less than 16,000 triangles, or a scene of less than 30 objects). We want to conclude that for large or complex scenes the dual PHANToM setup clearly has its limitations. In such a case, one can consider to afford the extra code complexity for a distributed setup, although network delay certainly will be a constraint in this case.

## 6 Acknowledgements

We first of all want to thank Geert Van De Boer for his coding effort in realizing this test and getting a dual PHANToM setup to work. Next, we also want to thank Jan van Erp en Hendrik-Jan van Veen from TNO (the Netherlands) for their suggestion to install Windows NT instead of Windows 2000.

## 7 References

[ACOST02] Scene Complexity: A measure for real-time stable haptic applications, E.Acosta, B.Temkin, October 2001, Aspen; CO, USA. 2001

[ALHAL01] Tele-Handshake: A Cooperative Shared Haptic Virtual Environment, M.O.Alhalabi, S.Horiguchi, EuroHaptics 2001, July 2001, Birmingham, UK.

[ANDER02] The ActivePolygon Polygonal Algorithm for Haptic Force Generation, T. Anderson, N. Brown, Novint Technologies, PUG2001, October 2001, Aspen; CO, USA. 2001

[DEBOE02] A networked two-handed haptic experience: The Virtual Percussionist, J. De Boeck, P. Vandoren, K. Coninx, VRIC2001, June 2002, Laval, France, pp 131-139

[HESPA00] Haptic Collaboration over the Internet, J. Hespana, M. McLaughlin, G. Sukhatme, M. Akbarian, R. Grag, W. Zhu, Proceedings of the fifth PHANToM Users Group Workshop; October 28-30, Aspen; CO, USA. 2000

[LOOP87] Smooth Subdivision Surfaces Based on Triangles, C. Loop, University of Utah, Department of Mathematics, Utah, US, August 1987

[RAYMA01] Fast Haptic Rendering of Complex Objects Using Subdivision Surfaces, C. Raymaekers, K. Beets; F. Van Reeth, PUG2001, October 2001, Aspen; CO, USA. 2001

# Response time consistency of the GHOST force loop

A. E. Kirkpatrick and Jason Sze
School of Computing Science
Simon Fraser University
Burnaby, BC, V5A 1S6 Canada
{ted,jszea}@cs.sfu.ca

## Abstract

The consistency with which Windows 2000 invokes the GHOST force loop was measured on a 900 MHz machine. The median loop time was accurate at 1 ms. However, as the computation in the force loop increased to 500 ms, the consistency of the loop decreased up to 25%. Inaccuracies in loop times were also introduced by higher network load.

## Introduction

The quality of the force display is an important factor in haptic applications[1]. The device, control algorithms, and application program all contribute to the quality of the display. All these components run in cooperation with a silent partner, the operating system kernel. The kernel provides the environment in which the device driver runs and the low-level facilities by which the driver communicates with the hardware. It provides the file system, network, interprocess communication, and virtual memory facilities upon which the application is based. Most importantly, the kernel scheduler determines when the application will refresh the force and graphic displays and the world model.

While much work has been done to locate and eliminate inadequacies in force display, control algorithm, and application design, we are not aware of any research on the effect of the operating system algorithms on the performance of haptic systems. To date, researchers and application programmers alike have presumed that the operating system is "good enough". In this paper, we consider the effects the operating system scheduler might have on this performance. We begin by considering the possible mechanisms by which the scheduler might impact system performance. We point out that an important measure of system performance should be the distribution of response times for the force refresh loop. We then present some initial measurements of the response time distribution for a particular PHANTOM configuration, a single-processor system running Windows 2000 and GHOST 3.1. We conclude with a discussion of the implications of these results for haptic applications.

---

[1] In a talk given at PUG '01, the first author emphasized the importance of separating our terms for display technologies—forces and graphics—from the haptic and visual systems, the human perceptual systems interpreting those displays. In keeping with that principle, we shall refer to "force displays", the "force rendering loop", and so forth. However, given the established usage of such broader terms as "haptic application" and "haptic programming", we retain them here.

## The operating system scheduler and the structure of a haptic application

Haptic programming is inherently multi-threaded. The classic structure for haptic applications consists of two loops, one computing the force display based upon the current cursor position and the location of objects in the world model, the second computing the graphic display based upon the same information. Because the human visual and tactual receptors have differing response rates, these loops run at different rates. Currently, SensAble's GHOST software architecture sets the force loop at 1000 Hz. Graphics loops typically run between 10 and 40 Hz. More importantly, because the consistency requirements of the tactual mechanoreceptors are more stringent than for the visual receptors, the force refresh loop is typically run at a higher priority than the graphics loop.

An important but little discussed consequence of this multithreaded architecture is that it makes the operating system an inherent component of the application, with operating system scheduling algorithms limiting the application's quality of service. The application (or, in the case of GHOST users, the application framework) may request a theoretical rate of force display but it is the scheduler that determines the actual rate. This scheduler is itself a complex algorithm, particularly when considered in terms of its interactions with the other services provided by the operating system. Consequently, it is imperative for haptic programmers to have clear descriptions of the limits of the scheduler they are using. In this paper, we begin developing such a description.

## The scheduler as a source of noise in the force signal

The fundamental force computation is a read-compute-write loop. For each tic of the clock, the application reads the current location of the cursor, computes forces at the tip based upon its interactions with nearby objects, then writes the forces back to the motors of the display. The scheduler determines when each iteration of the loop occurs.

There are two possible kinds of noise the scheduler can introduce into the force signal. First, it can invoke the force calculation consistently slower or faster than the stated rate. We refer to this effect as *drift*. Second, the time between successive invocations may vary. We call this effect *spread*. We note that some degree of drift and spread is inevitable. The important question is their magnitude.

Spread and drift have several consequences. Irregular invocations of the force loop can create irregularities in changes to the displayed force. If the application assumes time between loop iterations is constant, timing spread will produce inaccurate force computations. This is potentially most severe for algorithms that use higher-order terms such as velocity and acceleration. Longer delays between force computations can produce large force changes when the cursor is penetrating a rigid surface. Extreme force changes can cause the drivers to shut down the computation. Finally, irregular force computations can produce inaccurate high-frequency forces. In the worst case, small surface features may be omitted altogether.

The ultimate metric of the above effects is psychophysical: Does the human user perceive the irregularities in a given application? However, such a metric incorporates far more than the operating system effects. For the purpose of measuring the specific contribution of operating system delay, we instead directly timed the invocations of the force loop.

## Method

Timing loop consistency was measured using a skeletal GHOST application. The virtual world consisted of a single object located at the origin of the coordinate system. The object had a bounding volume far larger than the PHANTOM working area, causing its collisionDetect () method to be called for every invocation of the force loop. The collisionDetect () routine stored the time of its invocation in an array. Time was recorded using the Windows HighPerformanceTimer facility. This timer operates under ten μs accuracy, more than good enough for accurate measurement of the one ms force loop. The collisionDetect () routine optionally executed a busy delay loop. The delay simulated varying degrees of force computation. The collisionDetect () routine completed by returning zero, indicating no collision with the object.

The body of the application consisted of a graphics loop executed by a Windows timer interrupt every ten ms. This loop also recorded its execution time but this data is not reported here. The graphics loop displayed the PHANTOM cursor as an OpenGL sphere but did no other rendering.

This sample application was run for one minute. The PHANTOM was not touched during this time. As there were no objects for the tip to contact in this virtual world, moving the PHANTOM would have had no effect on the results. After the minute elapsed, the differences between the 60,000 successive force loop invocations were computed and written to a file.

Timings were collected under various conditions. In the unloaded condition, no other applications were active. In successive runs, the force loop delay was varied from 0 to 700 μs in increments of 100 μs, with a final run of 750 μs. In the loaded condition, a network-intensive application was run simultaneously. The network application opened up ten TCP connections and received an 888 byte packet from each connection every ms, for a total throughput of 8.9 Mb/s. This application created conditions of high network load. The GHOST timing application was run concurrently, again with force loop delays ranging from 0 to 750 μs.

All timing runs were performed on a Windows 2000 (release 5.00.2195, SP2) system with GHOST 3.1. The hardware was a single-processor AMD 900 MHz Thunderbird with 500 Mb of PC-133 SDRAM and an nVidia GeForce2 card.

## Results

To measure the extent of drift and spread, the .01, .05, .50, .95, and .99 quantiles were computed for the 59,999 elapsed loop times for each run. For all runs, the median (.50 quantile) was 1.00 ms. This configuration of Windows has effectively no drift under the tested conditions.

To .01, .05, .95, and .99 quantiles are a good measure of the spread of the loop times. The inner .05 to .95 band indicates the range of 90% of the loop times. A further 8% of the loop times lie in the outer band, outside the .05 -.95 range but within the .01 and .99 quantiles. Given a uniform distribution of these times, on average one loop every 80 ms (12.5 Hz) would fall in the outer 8% band. This frequency is within the detection range of human mechanoreceptors.

Figure 1 shows a plot of the .01, .05, .95, and .99 quantiles for the unloaded condition (solid lines) and the loaded condition (dashed lines). The 90% band is tight, ±.03 ms. While the 90% band is constant irrespective of the loop delay, the 95% band widens as the computation load of the haptic loop increases, from ±.08 ms at the 0 ms delay to a negatively skewed interval (0.66 to

1.18 ms) at the 500 ms delay. Beyond 500 ms, the lower limit of the loop time becomes bounded from below by the loop delay itself and the range shrinks, although the upper limit remains high.

The loaded condition has a wider spread than the unloaded condition but is much less affected by increasing loop delay. The 90% band is wider (±.08 ms) than for the unloaded condition. The 95% band is relatively constant with a more symmetric interval (.72 to 1.21 ms) of about the same width as the unloaded condition. Unlike the 95% band for the unloaded condition, however, the loaded band is basically constant across the range of delays, with only a slight decrease in the lower limit.

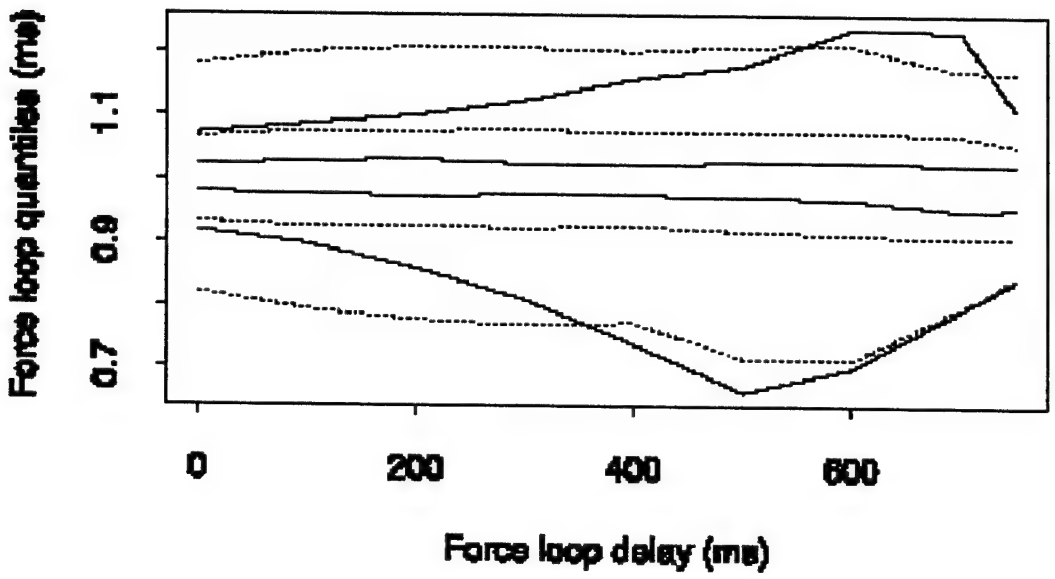



**Figure 1: .01 (lowest line), .05, .95, and .99 (highest line) quantiles for unloaded (solid lines) and loaded (dashed lines) conditions at various levels of delay.**

## Conclusions

This system configuration exhibited generally stable performance in the consistency of its force loop. However, even for conditions of minimal load, there was sufficient spread to the response times to have humanly perceptible effects. Developers of haptic applications with exacting performance standards may wish to account for instabilities introduced by the operating system scheduling algorithms. Of course, the range of haptic configurations is vast. The processor and GHOST release used in this initial study have both been superceded by more recent versions. Nonetheless, we suggest that the operating system scheduler will have potential impact on the performance of haptic applications for some time to come. We plan to extend this work to measure the performance of more elaborate configurations. We also plan to examine the psychophysical consequences of the variability introduced by the operating system.

**MONDAY**

8:30-9:00 AM Second Day Remarks -- Conference Chair(s)

9:00-10:00 AM Invited Speaker – Michael Wallace, Global Haptics, Inc.
Current Status of Haptics: Part II

10:00-10:15 AM Special Report IV – Results of Roundtable on Haptics Format

10:15-10:30 AM Break

10:30-12:00 PM Paper Session IV – Applications

1. A Testbed Haptic Particle-Spring-System Tool for Interactive Data manipulation, Chris Harding, Shell Geoscience Systems
2. Using Haptics Interaction in Bioinformatics Application – Arthurine Breckenridge, Sandia National Labs

12 Noon –1:00 PM Lunch

12 Noon – 2:00 PM Demo Room Open
Enjoy Bishop's Lodge/New Mexico Siesta

2:00 – 3:00 PM Paper Session V – Applications

1. Virtual Haptic Validation for Service Manual Generation – Christopher Volpe, Russell Blue, GE Global Research Center
2. Evaluation of the PHANToM Playback Capability – Robert L. Williams II, Mayank Srivastava, Robert Conatser, Jr, John Howell, Ohio University
3. A System for Accurate Collocation of Haptics and Graphics – Karl Reinig, Joshua Eskin, University of Colorado, Center for Human Simulations

3:00 – 3:15 PM Special Report V – Computer Haptics DVD 2003

3:15 – 3:30 PM Break/Demos

3:30 – 4:30 PM Paper Session VI – Performance

1. An Experimental Study of Performance and Presence in Heterogeneous Haptic Collaboration: Some Preliminary Findings – Wei Peng, Margaret McLaughlin, Gaurav Sukhatme, Weirong Zhu, Jacob Parks, University of Southern California
2. Comparison of Human Haptic Size Discrimination Performance in Real and Simulated Environments – Marcia O'Malley, Rice University

4:30 – 5:00 PM Special Report VI – Wrap-up

6:00 PM – 7:00 PM Dinner

6:00 PM – 10:00 PM Demo Room Open

**Invited Speaker – Michael Wallace**

Can a hand-held 3D input device that doesn't employ force-feedback be considered truly haptic? Can such a device outperform force-feedback devices? Could it complement such devices? Global Haptics thinks all of the above, at least in part. The history of this nanoscopic company is short but full of highlights, including the defeat of a major computer company in an IP-type battle. Mike Wallace, founder and President of Global Haptics, gives a summary of the 'haptic orb' technology and compares it (as best he can) to other force-feedback solutions. He'll apply his perspectives as an award-winning sculptor and earth scientist to how those fields are approached by haptics. He follows with his perceptions on the common hurdles that all haptics companies face in reaching a mass market.

# A Testbed Haptic Particle-Spring-System Tool for Interactive Data Manipulation

Chris Harding, Bluware Inc.

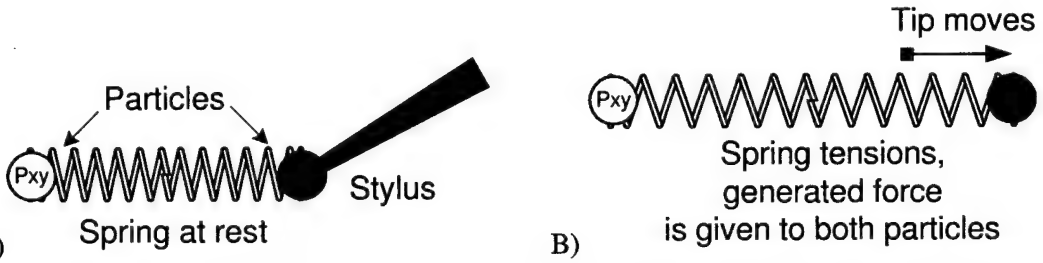
## INTRODUCTION

This paper describes my ongoing exploration of particle-spring systems for haptic interaction with 2D grid data. The work is based on earlier volume-haptics works (such as [1]) and similar to the recently presented work of [3] and [4]. The system simulates a network of interconnected particles and springs, each particle, with the exception of the phantom stylus' tip, is considered a mass in motion (but without extent) with its trajectory described by simple Newtonian physics equations (such as force = mass * acceleration). Springs connect these particles to each other and to the tip and redistribute the forces among them. In addition to inertia, an underlying 2D-grid of 8-bit values influences the particles locally as they travel over it. At each point on the grid, a "density" value and its gradient is used to deflect or slow down the particle.

The tool is very configurable can be used change the underlying data grid by selecting the tool's size and choosing a type of tool-action to either increase or decrease the data value in some fashion. On the click on the stylus' button the action is performed on the data within the tool's area and the haptic feedback will reflect this change in data through changes in force within the particle-spring system. Through the particles (and their contribution to the overall force applied to the tip) the user is able to feel the underlying data values. The tip is a special particle in that the user determines the current position and in turn receives the net force-feedback resulting from the simulation of the network.

## BASIC PARTICLE-SPRING PHYSICS

I will first talk about a simple single-particle-single-spring-system: a proxy-particle (called Pxy the sketch below) is connected to the tip-particle (called Tip) through a spring - the phantom's stylus is of course just an extension of the tip. In timestep 0 (A) the spring's length is exactly its at its so-called restlength and no forces are applied to either particle:




I will walk step-by-step through a single haptic simulation cycle. Assume that at the start of timestep 1 the user has moved the tip a bit to the right (B). The spring's extent is the distance between tip and proxy and the two particles move with a certain relative velocity (in the following equations vectors start with upper-cases, scalars start with lower-cases):

$$\text{Distance}_{\text{proxy to tip}} = \text{Pos}_{\text{proxy}} - \text{Pos}_{\text{tip}} \quad \{1\}$$

$$\text{Velocity}_{\text{relative}} = \text{Velocity}_{\text{proxy}} - \text{Velocity}_{\text{tip}} \quad \{2\}$$

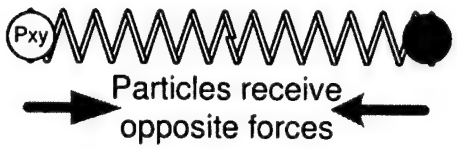
The spring is characterized by three parameters: a spring-stiffness constant $k_{\text{stiffness}}$, a dampening constant $k_{\text{dampen}}$ and a restlength. The force exerted on the now extended spring is described by Hooke's law:

$$F_{\text{stiffnes-only spring}} = k_{\text{stiffness}} * (\text{distance}_{\text{proxy to tip}} - \text{restlength}) * \text{Direction}_{\text{proxy to tip}} \quad \{3\}$$

$\text{Direction}_{\text{proxy to tip}}$ is a unit vector of $\text{Distance}_{\text{proxy to tip}}$ (length 1.0) while $\text{distance}_{\text{proxy to tip}}$ is its length. As it stands now, energy introduced into the system by moving the tip around, would never be lost, in order to stabilize the system I need to dampen the spring, which involves another constant and is proportional to the relative velocity of the two particles:

$$F_{\text{dampened spring}} = F_{\text{stiffness-only spring}} - k_{\text{dampen}} * \text{Velocity}_{\text{relative}} \quad \{4\}$$

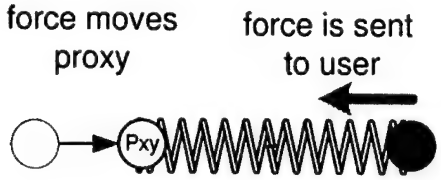
Note that I will not deal with torque in this simulation, as the Desktop phantom is unable to produce torque. Now that I know the force currently "contained" in the spring, its easy to get the forces applied to its two particles – one will be pulled in one direction, the other in the opposite direction, both with the same amount of spring force:




$$F_{tip} = -F_{dampened\ spring} \quad \{5\}$$

$$F_{proxy} = F_{dampened\ spring} \quad \{6\}$$

Now that each particle has a force applied it's very simple to figure out where it will move to (essentially its trajectory). I get the exact amount of time passed since the last haptic timestep (or frame) from the PHANToM. With this timestep (dt) and force (F) known and with the mass m already assigned I can integrate over time and get the particle's change in velocity (dV){eq.7–9}. I'm using the very simple Euler integration scheme, for a more in-depth look at these integration issues see [2], pp.173 ff. Once I know the particle's velocity (and, of course position) in the last timestep, I can get the "new" velocity (i.e., the particle's velocity after dt has passed) {eq.10}. Having the new velocity, I can now calculate the particle's new position by adding its change in position, which I get again by integrating over time {eq.11}:




$$F = m * a \quad \{7\}$$

$$F = m * dV/dt \quad \{8\}$$

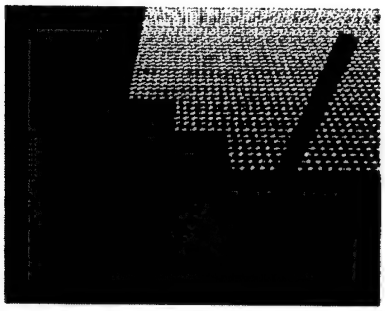
$$dV = (F/m) * dt \quad \{9\}$$

$$Velocity_{new} = Velocity_{current} + dV \quad \{10\}$$

$$Pos_{new} = Pos_{current} + Velocity_{new} * dt \quad \{11\}$$

I now move the proxy to its new position and in the next haptic frame (timestep 2) I start over by again calculating the force contained in the spring (which has changed as the distance between its particles has changed) and again figure out what the proxy's new position is going to be, etc. The procedure above is shortened for the tip as its position is given as input by the user, so I simply need send the (accumulated) force back to the user.

## DRAG, STATIC FRICTION AND GRADIENT FORCE



Although its fun to play with different weights for the proxy and different spring-length, stiffness and dampening values, the real goal is to feel the underlying data and ultimately manipulate it. For this purpose I set the spring's restlength to 0 and add forces derived from static friction, drag and gradients to the proxy's momentum. For this test project I assume the data I move through is a 2D regular grid of 8-bit density values. I have experimented two ways of getting density values at any point on the 2D grid. One way simply takes the value of the nearest grid center, the other way does a linear interpolation by dividing a grid cell into 2 triangles and getting the barycentric coordinate of the triangle it's in.

Although the data is originally in 8-bit I normalize it to 0.0 to 1.0 (i.e., 0.0 = 0; 1.0 = 255). Graphically the data is shown as a sort of checkerboard on the screen with the stylus connected to the proxy with a spring (here show as wireframe for clarity and with a very weak spring)

### Drag

The concept of drag is similar to dampening - data-dependent drag can be easy implemented by subtracting a small percentage from the particles velocity. The amount of drag subtracted is of course dependent on the density value at the particle's current position i.e., denser materials subtract more velocity and cause higher drag. For example, I could reduce the velocity to only 95% of the nominal value for a density of 1.0 and subtract nothing when the density is 0.0 (the drag constant is set to 0.05, meaning 5%):

$$Velocity_{with\ drag} = Velocity_{new} * (1.0 - density\ @\ Pos_{proxy} * constant_{drag}) \quad \{10a\}$$

As a result the drag on the proxy now varies depending on the underlying density at the proxy's position. As the tip is moved through higher density parts of the data, the user feels these variations of the drag as a higher resistance to overcome.

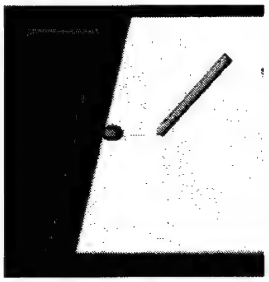
### Gradient Force

Another way the proxy is influenced by data is via the data's gradient. In contrast to the two friction-related techniques, this is a somewhat artificial concept as it actually adds energy into the system It is however very helpful in better defining material boundaries which is why I have included it.

Getting a gradient at a certain point on the 2D grid means getting the differences in values over a certain distance and for each dimension. I've implemented simple central differences, a 3x3 Sobel filter and a 5x5 "circular" filter – the physical extent of the gradient kernel is kept variable which allows me to fine-tune the tool-size later. The gradient force can be calculated in two ways, a) the gradient vectors 's direction **and** magnitude is given by the difference in data or b) the gradient's magnitude could be made dependent on the (normalized) density value at the proxy's position:

$$F_{proxy} \mathrel{+}= \text{Gradient} @ \text{Pos}_{proxy} * \text{constant}_{gradient} \quad \{6a\}$$

$$F_{proxy} \mathrel{+}= \text{Gradient} @ \text{Pos}_{proxy} * \text{density} @ \text{Pos}_{proxy} * \text{constant}_{gradient} \quad \{6a\text{-alternative}\}$$



The image shows a live case of the proxy having been dragged from zero density material (black on the left) into high-density material (white) The tip is at the end of the stylus to the right and the (very weak) spring connecting proxy and tip is drawn as a line in between (with decent spring stiffness, this distance is very short). I have set the gradient kernels size to coincide with the sphere. The gradient (shown by a red debug vector going left from the proxy) is just between the last of the black material on the right and the white material on then left – this is the width of the "skin" zone. Any further pull to the right would result in the proxy "popping" through the skin after which drag will take over.

One of the trickier points in this type of simulation is to configure the system so that the transition between the skin area and the inside of a higher-density body is fairly smooth in other words that the change from gradient to drag does not result in too much of a "popping through" experience. The factors to tweak for this issue is obviously the gradient constant and the drag constant but also the size of the gradient kernel.

**Static Friction**

Another type of friction that's easy to incorporate is static friction. I demand that the particle has a certain amount of force applied to it before I start moving it. In other words after {6a} there needs to be a check:

$$\textbf{if } |F_{proxy}| < \text{density} * \text{constant}_{static\ friction} \ \textbf{don't move} \quad \{6b\}$$

This will leave the proxy stuck until the tip pulls at it with a force large enough to move it. Even after the proxy gets going it may still fall into a stop-and-go pattern that feels a lot like moving a finger over rubber

**Units and constants used**

I want talk a bit about units used in this project and roughly what values the various constants one can expect to work (or at least use as a safe start). The units the phantom arms moves in are in mm (m/1000), in my case the timesteps are in ms (s/1000) and the weight of the proxy is in gr (kg/1000). I found a mass of 50 gr to 200 gr worked for me, the spring constants were: 0.1 - 1.2 (gr/ms2) for the spring stiffness and 0.001 - 0.1 (gr/ms) for the spring dampening. For drag, 5% is my upper limit, more tends to overload the phantom in density 1.0 material. The gradient constant needs to be adjusted to the users needs, if its desirable to make shapes very hard the constant may need to be as high as 3.0

**Restpoint**

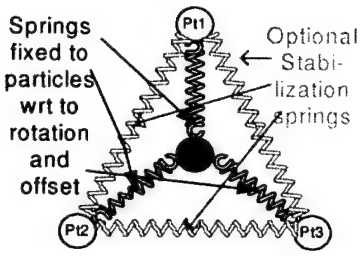
While it is possible to craft an interesting tool with the current spring model (with only a rest length) I found it necessary to "pin down" one end of the spring in relation the other end. The restpoint is a vector that defines the desired local offset from the proxy to the tip. I want to also incorporate the rotation of the stylus around the tip even if the Desktop phantom used here cannot project torque back to the user and there are some issues with "faking" torque. The stylus' rotation matrix is again given by the PHANToM, all I have to do is to multiply the restpoint (vector) with it to get the rotated restpoint (vector):

$$\text{Restpoint}_{after\ Rotation} = \text{stylus-rotation-matrix} * \text{Restpoint} \quad \{3a\}$$

$$F_{stiffnes\text{-}only\ spring} = k_{stiffness} * (\text{Distance}_{proxy\ to\ tip} - \text{Restpoint}_{after\ Rotation}) \quad \{3b\}$$

## COMBINING PARTICLES AND SPRINGS INTO A NETWORK

Now that I can offset and rotate the proxy round the tip I can build a network where the tip is connected via 3 different springs to the (proxy) particles and they are connected to each other via 3 stabilization springs.

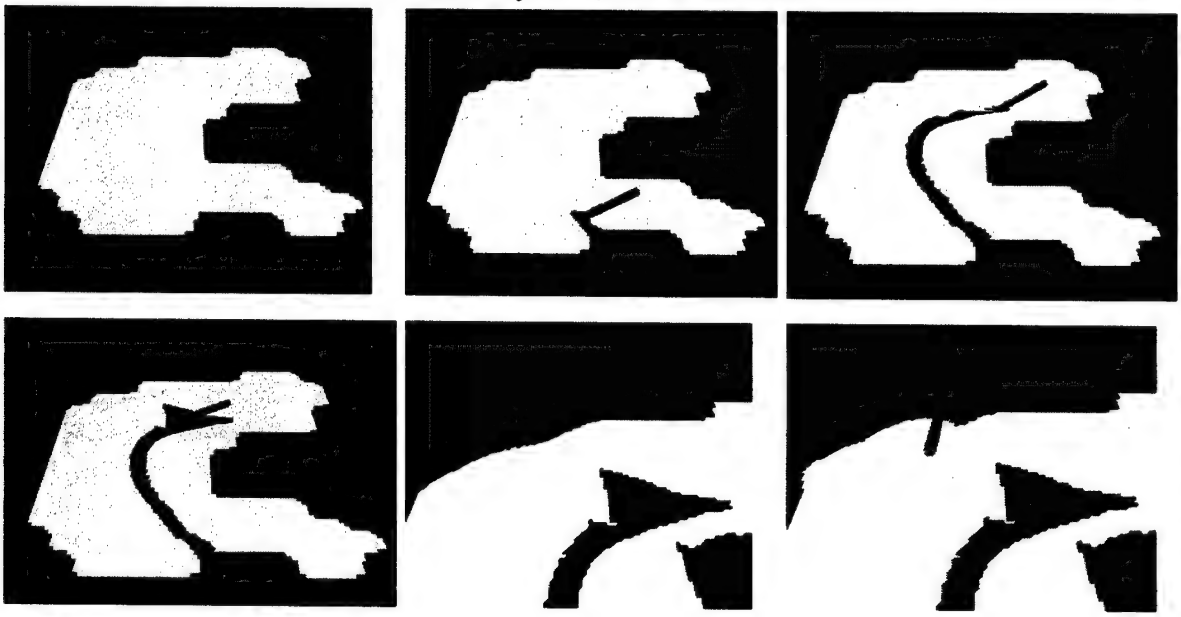



The physics simulation now needs to take care of 6 springs and 4 particles, i.e., get the force for each spring, which is in turn dependent on the position, velocity, etc. of its two particles and apply this force to each particle. Because a particle could get its (partial) force from more then one spring, I need to reset each particle's force before I start the accumulation. After each particle has its total force I again integrate over the time passed since the last haptic frame and get its velocity and displacement, the total force for the tip is sent off to the user. The structure of the network is given by the restpoint of the 3 springs, the could also be connections between the particles via stabilization springs which are only distance constrained and can be used to increase overall stiffness of the tool-head.

## DATA MANIPULATION

The haptic tool is define by two qualities, it's size (extent) and its action i.e., what alterations it performs on the underlying data. When the user presses the button at the phantom's stylus, the data lying "inside" the tool is changed according to the type of action the user has selected. For the particle-spring network in the shape of a triangle, the inside is simply defined as the current area of the triangle. For a single particle (proxy) tool, I define a sphere with the size of the gradient kernel as its radius. As my data model defines density (essentially greyscale) values, I have primarily worked with simply adding and subtracting density.

As I have shown, the particles of the toolhead are influenced in various ways by the underlying data (drag, friction, gradient-repulsion), so a change in data will immediately affect the force-feedback to the user via the change in particle behavior. However, in order for this feedback to be smooth, the change of data has to occur in the haptic loop, increasing the number of computations done per haptic frame. The multi-particle case requires more complicated calculations to find the actual inside cells (involving a series of half-space calculations) while in the proxy-only the inside is define by its radius alone. The current system offers a multitude of factors that can be taken into account and that further determine the tool's behavior in a wider context. I have experimented with a variety of additional properties and constrains, such as tool size, tools strength, feathering (different strength at different parts within the tool) and taking the tool's velocity, density and gradient into account. Some snapshots of deleting and adding parts of a simple binary 200 x 200 pixel "playground" with a single while (density = 1.0) objects surrounded by black material (density = 0.0) are shown below:



Carving a path with a triangular tool. The path's width is influenced by the tool's orientation

Using the flat side to carve a cavity and round off a side.

Adding material onto the side

## REFERENCES

[1] Avila, R. S.,Sobierajski, L.M. (1996), A Haptic Interaction Method for Volume Visualization, Proc. IEEE Visualization '96, p. 76-85.

[2] Bourg, D.M. (2002), Physics for Game Developers, O'Reilly, ISBN 0-596-00006-5

[3] Lundin, K., Ynnerman, A, Gudmundsson, B. (2002), Proxy-based Haptic Feedback from Volumetric Density Data, Proc. Eurohaptics 2002, p. 104 – 109.

[4] Lundin, K. (2001), Natural Haptic Feedback from Volumetric Density Data, Technical Report LITH-ITN-EX-MT-5-SE, Dept. of Science and Technology, Linköping University

# Using Haptics Interaction in Bioinformatics Application

Arthurine Breckenridge
arbreck@sandia.gov

Ben Hamlet
bhamlet@nmt.edu

Sandia National Laboratories, New Mexico is continuing to research 3D user interfaces specifically with the addition of haptics, the sense of touch. Our project is researching computational and collaboration tools independent of content area. However, applying the work to specific problems is very beneficial guide to our development. The application specific content focus of this paper is a bioinformatics application.

Bioinformatics is the combination of computer science and biology (the youngest of the natural sciences). Sequence-structure-function prediction refers to the idea that, given a molecule's sequence identity, we would like to predict its three-dimensional structure and, from that structure, infer its molecular function. Researchers have uncovered reasonable evidence indicating that a protein's structure approximately determines its molecular functions, such as catalysis, DNA binding, and cell component binding.

Many results in experimental biology first appear in image form – a photo of an organism, cells, gels, or micro-array scans. As the quantity of these results accelerates, automatic extraction features and meaning from experimental images becomes critical. At the other end of the data pipeline, naïve 2D or 3D visualizations alone are inadequate for exploring bioinformatics data. Biologists need a visual environment that facilitates exploring high-dimensional data dependent on many parameters. Therefore, visualization of bioinformatics is a vast field. Our specific focus is highlighted in special interest box: intron/exon splicing.

**Basic Simulation Elements**

The two basic elements of a computer haptics simulation are graphics and force that actually work independently of each other. This divides the simulation into two distinct parts, each of which present unique problems that can be addressed in independent and different ways. Although a number of factors go into making a successful simulation, the success of ones application will ultimately rely on good refresh rates (i.e., +30 Hz for graphics and 1000 Hz for force feedback). Both graphics and haptics interaction can apply to the application content and the graphic/haptic user interface. The application will use e-Touch [tm] originally developed within Sandia National Laboratories, New Mexico and now licensed by Novint Technologies, Inc.

**Graphic/Haptic User Interface**

Force feedback has been implemented into the e-Touch [tm] user interface. The basic elements are documented on the www.novint.com website. Features such as craft navigation and menus with touch sensations are standard. In addition, this application has added 3D box selection of elements and magnetic pointer to start location. The start location pointer can be used as intelligent alignment tool.

**Graphics Generation**

A number of approaches are available for rendering molecular graphics. One is a simple shape like line, sphere, etc. Although simple to code, as the number of primitives increase, the graphics refresh rate decreases substantially. Quick fixes are to sort the data by atom type to avoid graphics transition states like color and material properties. Keep in mind that data pre-processing is highly desirable for all haptics simulations due to the relentless refresh rates that such an application demands. Another approach is to calculate which surfaces are hidden. Although removing hidden surfaces is substantially

faster than drawing them, when dealing with a large number of atoms, it can still exceed the haptics servo loop requirements. Of course, the classic solution is to draw different levels of details based on screen position, user selection via graphical user interface, and spatial decomposition

Another graphical method used that seems promising is to fool the eye that it is seeing a 3D object since the sense of touch sends another signal that the object is actually has

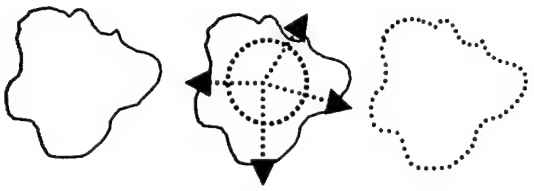


Figure 1. Circular Vector Field

shape (i.e., round atom). In this method, one determines the shape and contour of the protein's surface within a given level of detail and creates a graphics triangular surface mesh. For example, knowing the center location of each atom, the furthest sphere intercept along these vectors can be calculated (Figure 1).

Each point of intersection will then be used as a vertex in creating a surface mesh of triangles. The surface mesh can be rendered on a PC cluster since it requires an entire program to simply plot it. Once the coordinates of the surface points have been calculated, there is still a substantial amount of compute cycles needed to create a texture map for the protein and define the shadowing of the polygons. If it seems complex, this graphics solution is actually beneficial to maintain the needed haptics refresh rates and scale linearly.

**Force Calculations**

Computer haptics using the PHANToM is basically surface forces that act like springs, where the force exerted is equal to a constant times the depth of penetration, F= kx. Because force magnitude is dependent on the depth of penetration, when one touches two spheres at the same time but does not penetrate their surfaces an equal distance, one surface will exert a greater force than the other (Figure 2).

All of this takes place at 1000 Hz, which causes a high frequency oscillation of the PHANToM that can be felt or even heard in many circumstances. Force buzzing is simply where the PHANToM vibrates because of an inconsistent or unstable stream of forces is being fed to it. Object springiness can cause this because when two identical spheres push on the cursor, and the first sphere pushes harder than the second, the cursor will penetrate the second sphere more deeply, inducing a strong force in that sphere and so on. There are a number of solutions to this problem, many of which are not simple and may have undesirable side effects. The easiest way to eliminate buzzing, at least to the extend of audible recognition, is to vary the spring constant of the surface forces proportionally to the number of objects being touched. For example, if one begins to touch one atom, you will feel the maximum force. Then if one moves the cursor to touch more atoms, the force will decrease (.05 to .1) and will increase as you touch less. If one's range of forces and the speed at which one ramps them is reasonable, this method can alleviate solid surface buzzing.

Another solution to force buzzing produced by overlapping objects is to not have overlapping objects. Much in the same way that a graphics mesh of a protein molecule can be generated, a force shell for a protein can be created. By

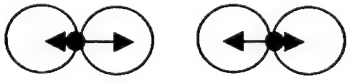
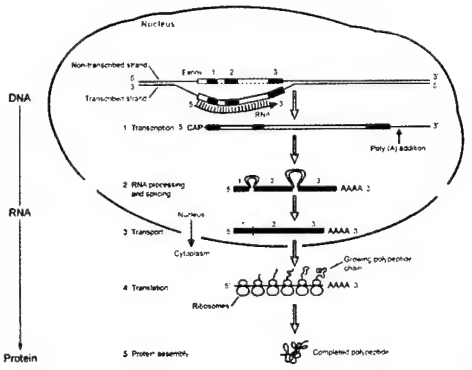


Figure 2. Force Buzzing

pre-processing the normals, one no longer has to worry about inter-object forces. A force shell will have the affect of making your protein completely impenetrable and may greatly reduce the value of one's simulation. Whereas the graphics shell still seemed 3D due to the force representation, this feature is lost if both graphics and force shells are used.

## Special Interest: Intron/Exon splicing

The creation of life begins with a single cell that divides into two. And, why does this process sometimes malfunction, leading to defects, cancers, and other diseases? And, could this process sometimes function as planned and could potentially be used by terrorists. At the brink of the twenty-first century, there are 24 complete "draft" genomes available in public databases including that of the human genome. Though impressive information, the real challenge is transforming the torrent of raw data into biological knowledge. Thus, bioinformatics, the combination of biology and computer science is introduced. When asked what is the Holy Grail of bioinformatics, most researchers would answer the ultimate goal of a genome project is to determine the function and biological role for all of the genes of interest ideally in silico.




```
0020000   002310  045062  006520  436001  660023  023005  006001  000000
0020020   024001  100121  032000  001110  000200  010304  065000  000000
0020040   034100  000000  023001  007760  000000  014000  004012  166007
0020060   701032  000100  020400  000000  321000  057000  000000  023310
0020100   000000  000000  000001  111100  234000  221500  020020  000100
0020140   003427  000000  000000  023100  066010  000000  100777  004000
0020160   000250  020000  001000  430056  256000  000000  000000  000010
0020200   021450  001000  000000  110010  000000  000000  000000  003042
0020220   550210  034011  002200  023570  000010  000000  000000  200543
```

(a) An Octal Dump

```
1021 GCTGAAGGTC GCCTCGTCGC GGGTGTAACG CAGACCGCCC GACAGGGTCA GCTTGTCGGT
1081 CAGGTCATAG TCGACCTGGC CGAACACGGC GAAGCTGTTG CGCGTCTGCT TGTGCAGGTT
1141 GTCGTACTGG GTCCGGGCCG TATAGCTGGT CCAGTGGGTG TCGAAGTCGA TATCGTCCGA
1201 GACGTAGAAG GCGCCCACGA TCCAGTCCAG ACGACCGTCG TTGTCGTCGC TCAAACGCAT
1261 TTCCAGCGAG GTCTGGTTCA GCGTCTCATC GGCATCGAAG CGCGAGGCCG GATAGGGCGA
1321 GCCGTCGCCG TTGCCCTGCA CCGAGCGAGT CGTGCCCTGG ACGTTGCCGA TCACCGCCAG
1381 ATTCAGGTCA TCGCCCAGAC GATGCTGCAA CGCCGCCGAC ACGCCATAGA TGTCGATGCT
```

(b) A DNA String Extracted from GenBank

Figure A. Central Dogma as seen by a Biologist on the Right and a Computer Scientist on the Left

Though considerable progress has been made in understanding flow of information known as the "central dogma, Biology is the youngest of the natural sciences. Figure A.a) shows an octal dump of an assembly language code, which once upon a time ubiquitous in debugging computer programs, which nowadays become a rarity. Figure A.b) show a portion of the GenBank database presenting parts of DNA containing the letters A,C,G,T suggesting a quaternary kind of dump. Hopefully, this important but initial analysis will also become a rarity, as we understand more structural dynamics.

Because of the interdependent flow of information represented by the central dogma, one can begin discussion of the molecular expression of genetics of gene expression at any of its three informational levels: DNA, RNA, or protein. In 1953, James Watson and Francis Crick deduced the secondary structure of DNA. This was one of the most important biological advances, since it led to an understanding of the relationship of the DNA structure to its function, particularly to the way it was replicated. Scientists now are beginning to understand the process of copying DNA into RNA called transcription. The perceived role of RNA has changed from a passive messenger of information and scaffold for proteins to a central and active role in the functioning of the cell. To understand how a specific RNA molecule operates, its functional structure need to be better understood. It may be deduced from (costly and difficult to obtain) X-ray diffraction or NMR data only for short RNAs. In most cases, however, only the single RNA sequence (the primary structure) without further information regarding its functional form is available.

The secondary structure differs from the Watson-Crick DNA secondary structure in that it is generally single-stranded. When left in its environment, this molecule will fold itself into its secondary structure by creating base-pairs (an A with a U, a C with a G, or even a U with a G). Scientists are now deducing the secondary structure of RNA by using difference equations and dynamic programming. In 1977, it was determined that genes of higher organisms did not follow the simplest "central dogma" model. Instead, few genes exist as continuous coding sequences. Rather, one or more non-coding regions interrupt the vast majority of genes. These intervening sequences, called introns, are initially transcribed into pre-mRNA in the nucleus but are not present in the mature mRNA in the cytoplasm. Introns alternate with coding sequences, or exons, that ultimately encodes the amino acid sequence of the proteins. The portions corresponding to introns are removed, and the segments corresponding to exons are spliced together. The production of mRNA has to occur in the correct amount, in the correct place, and at the correct time during development or during the cell cycle. Each of the steps in this complex pathway is prone to error, and mutations that interfere with the individual steps have been implicated in a number of inherited genetic disorders.

Considering human genes contain ten times more intronic than exonic sequence, this weakness in the understanding of higher eukaryotes genetics is becoming increasingly apparent. Recent bioinformatic studies suggest that at least one third of human genes are alternatively spliced. Intron's evolutionary functionality is in much debate. However, there is agreement that the splicing sequences are neither strong nor unique enough signals since the splice sequence can be found in other parts of the mRNA. In the context of computer prediction of exon boundaries based only on primary sequences, this makes the task quite difficult. Therefore, what factors in the nucleus may also facilitate identification of sites? Shown in Figure A, this proposal is interested in process for detecting nuclear introns. Soon after discovery of splicing it became evident that this process also occurred in lower eukaryotes like yeast. Because the chemical mechanism of splicing and many factors involved in the splicing process are conserved from yeast to man, the budding yeast, Saccharomyces cerevisiae, has become an important model systems for the analysis of splicing. A comprehensive study of splicing in vivo of yeast (Spingola, 1999 ) concluded that results show that correct prediction of introns remains a significant barrier to understanding the structure, function and coding capacity of eukaryotic genomes, even in a supposedly simple system like yeast. As complete eukaryotic genome sequences become available, better methods for predicting RNA splicing signals in raw sequence will be necessary in order to discover genes and predict their expression.

Why study secondary structure? One of the major problems facing computational biology is the fact that sequence information is available in far greater quantities than information about the three-dimensional structure. While the prediction of 3D RNA structures is unfeasible at present, the prediction of secondary structures is in principle tractable. In RNA the secondary structure elements are significantly more stable and form faster than the tertiary interactions. Thus, a separation of an RNA folding model into secondary (properly nested base pairs), and tertiary (non-planar nucleotide contacts) seems feasible. Determining the secondary structure of an RNA molecule is widely seen as a first step towards understanding its biological function.

Spingola, M, Grate, L., Haussler, D., and Ares, M., "Genome-wide bioinformatic and molecular analysis of introns in Saccharomyces cerevisiae", *RNA*, 5:221-234, 1999.

# Virtual Haptic Validation for Service Manual Generation

Christopher Volpe
volpecr@research.ge.com

Russell Blue
blue@research.ge.com

GE Global Research Center

## Abstract

We've designed and implemented a virtual validation system using haptics as well as other typical Virtual Reality equipment (head-mounted display, data-glove, and tracking hardware) for validating maintenance procedures and doing maintenance studies. The system uses a compact, sampled data representation to handle complicated geometric environments. Thus, the rate of collision detection is independent of the number of polygons in the source data. Also key to the system is an independent representation of the parts in the environment, allowing a random-access removal order. This allows a maintainer to validate a sequence of steps in a maintenance task, or instead to determine the sequence of steps defining the maintenance task. The validation system is part of a larger system (Service Manual Generation) to produce maintenance procedures in an automated fashion.

## Introduction

Maintainability analysis involves analyzing the design of a system to ensure maintenance actions can be performed quickly, safely and accurately. Complex systems such as aircraft, power systems and appliances require timely, reliable and accurate documentation to support maintenance. As a result of such complexities, technical manuals with textual descriptions and exploded views of equipment describing, for example, how to remove, inspect, repair and install parts, are required to keep many complex systems in working order. The documentation is an integral part of the maintenance process. Thus, a thorough maintainability analysis should include the analysis of the product design and documentation working together. Unfortunately, maintenance manuals are typically one of the last items developed and delivered in a complex system [1,2].

To address this problem, a research effort was begun under contract with the Air Force Research Lab's Human Effectiveness Directorate [3] to investigate an automated approach to maintenance manual development. The three year program, entitled "Service Manual Generation" (SMG) [2,4,5], was started in 2001 and consists of three main components: Sequence Generation (SG), Task Generation (TG), and Virtual Validation (VV). The SG component is responsible for analyzing the CAD geometry of the relevant parts, producing an exploded view of the assembly, and determining a sequence of part removals for the purpose of maintaining a particular part, such as a "Line Removable Unit" (LRU) of an aircraft engine. The TG component takes the output of the SG component, and generates content for a maintenance manual authoring system, including the human-readable steps to perform the task, and Warnings, Notes, & Cautions associated with any of the steps. The final component, VV, has two main responsibilities. First, it enables the validation of the removal sequence produced by the SG component, to ensure that the sequence itself is feasible by performing the tasks after being given a visual depiction of the procedure. Second, it allows the user to verify the instructions produced by the TG component to ensure that the instructions adequately describe the task.

The VV component combines a number of virtual-reality technologies in an effort to provide a realistic environment for determining the validity of the steps within the sequence. These technologies include a stereo Helmet-Mounted Display (HMD) to immerse the user in the environment, a data-glove for modeling the hand in the removal process, a tracking system for mapping the real-world positions of the HMD and data glove to positions in the virtual environment, and a 6DOF SensAble ™ PHANToM™ for enabling realistic human interaction with the virtual environment (Figure 1). It is the use of haptics in this virtual environment that is the main topic of this paper.

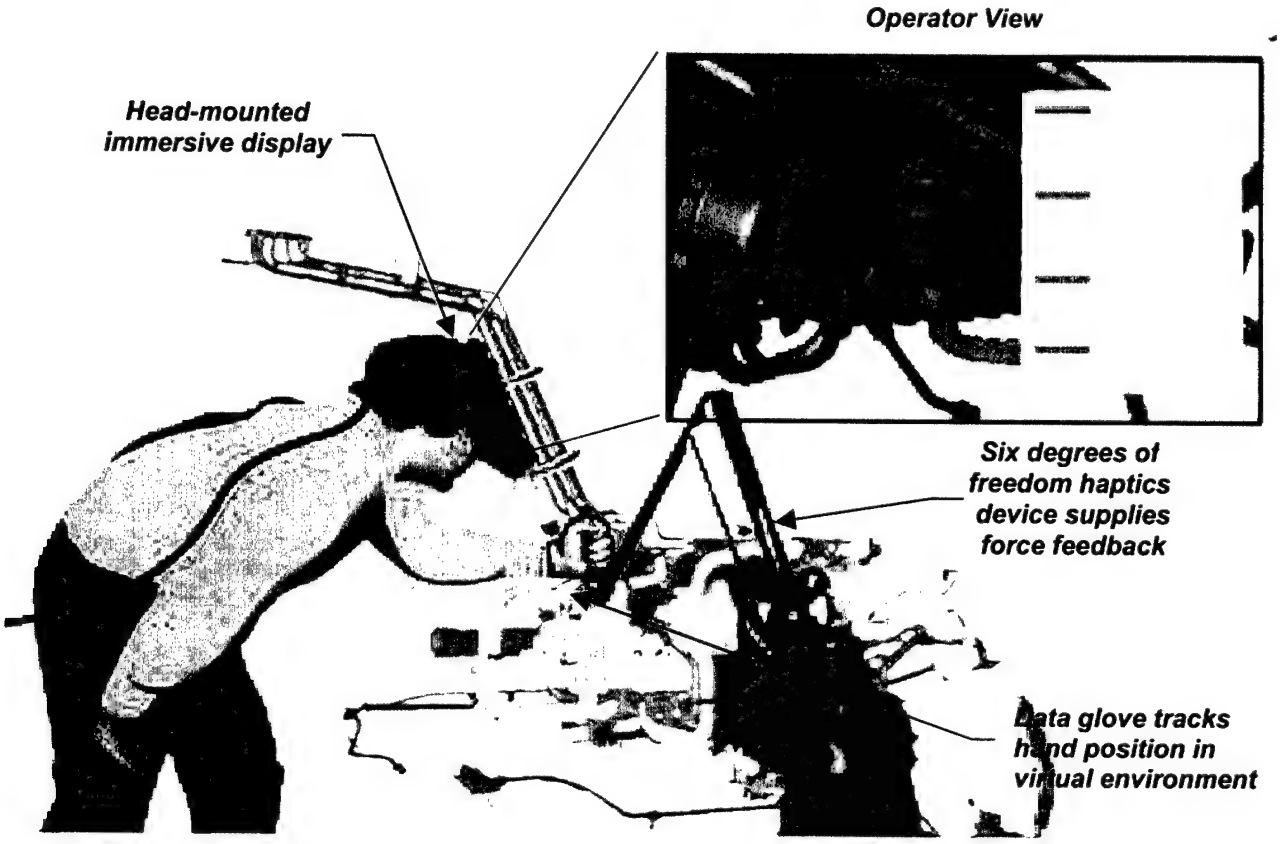



Figure 1: VV system

## Haptics for Virtual Validation

GE Global Research has been working for some time on the use of haptics for maintainability analysis. Initial investigation for this task began in 1998, when we developed a collision technique based on using points sampled from the surface of a moving part, and a volumetric grid containing proximity/penetration information for the non-moving parts, similar to the technique developed around the same time by Boeing [6]. In the fall of 1999, we delivered a prototype Haptic Path Planner tool to the Joint Strike Fighter (JSF) military engine program at GE Aircraft Engines. This work provided the foundation for the haptic component of the Virtual Validation system in SMG. For Virtual Validation, we needed to enhance our approach to allow for parts to be selected in a random (user-determined) order as the moving part for removal. This allows multiple parts to be haptically removed in rapid succession. In the paragraphs below, we describe in some detail how we represent data in our haptic environment in a way that allows for multiple part removal, and how we perform fast collision detection, independent of the polygonal complexity of the models used in the environment.

### Data Representation

There are three main components to our haptic representation. We describe each of these components in turn.

*Surface Points.* For each part that participates in the removal process, we generate a set of points that lie on the surface of the part which are evenly spaced and of sufficient density to adequately represent the part at a desired resolution. The surface points for a given part are used only when that part is the "moving part" in the environment.

*Penetration Map.* For each part relevant to the maintenance task, whether it is one to be removed or not, we generate a fine-resolution volumetric grid called the "penetration map", which contains proximity (penetration) information for points that lie outside (inside) the surface of the part, along with surface normals corresponding to the surface polygon nearest to the given point in the volumetric grid. In order to conserve space, this grid is arranged in a two level hierarchy in which a 4x4x4 block of grid points are grouped together to form a "brick". If all the grid points within a brick are greater than some maximum distance outside the part, or greater than some maximum distance inside the part, then the brick consists of a single value identifying that state. Otherwise, the brick is broken down into the 64 grid points with detailed penetration and surface normal information.

*Part Map.* The collection of (possibly overlapping) penetration maps of all the relevant parts defines a volume of space consisting of the union of all the volumes of the individual penetration maps. This combined volume is subdivided at a coarser resolution than the individual penetration maps to produce a grid called the "part map". This part map contains, at each grid location, a list of which parts are relevant within the space represented by the given part map grid location (Figure 2).

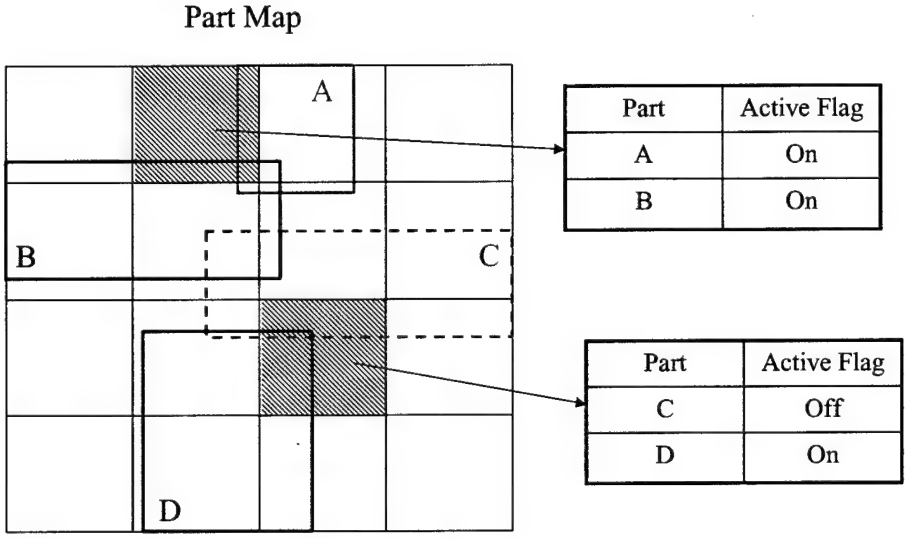



Figure 2: Part Map example

## Fast Collision Detection

At each iteration of the haptic collision detection process, the sampled points of the moving part are transformed according to the moving part's position and orientation in the virtual environment. The resulting location is indexed into the part map to determine which parts are possibly in contact with the given point. Since any one of these parts may have been previously removed in an earlier step of the removal sequence, each part has associated with it an "active" status flag, indicating whether or not it participates in collisions with the moving part (Figure 2).

For each active part in the part list at the current sampled point's location, we conduct a penetration test with that part's penetration map. Based on the sampled point's location, we look up the eight surrounding grid points in the penetration map and tri-linearly interpolate both the proximity/penetration distance and the associated surface normals. We keep track of whichever part the sampled point penetrates the most, and use that part's interpolated penetration distance and surface normal to calculate a collision response.

In order to take advantage of a sufficiently dense set of sampled points without wasting precious CPU cycles on points that have no chance of being in collision, we take advantage of the temporal coherence of the moving part through a technique called "skip counts". Each time a collision test is performed on a sampled point, we determine,

based on the distance of that point from the nearest surface and the maximum speed with which we allow a part to move, the minimum number of iterations that must pass before that point can possibly collide with any surface. This number becomes the "skip count" for the given point. At each iteration of the haptic loop, we decrement the skip count of each point for which the skip-count is non-zero, and perform the collision test only for those points whose skip-count is zero.

## Current Research Issues and Future Tasks

In order to make our system more robust, we are currently investigating a variety of techniques for dealing with frequently occurring pathological situations in real-world data. One such situation occurs when the part has regions where the thickness is on the order of our sampling resolution. We frequently see this case in aircraft engine data for hollow tubes where the wall of the tube is thin. When this occurs, adjacent penetration map elements may reside on either side of the wall, resulting in the ability to pass through a wall without ever landing within the interior material, and thus failing to detect the penetration. The result is that the part can erroneously move through some regions of the penetration maps. We are currently evaluating several candidate solutions to this problem.

In order to accurately model the affects of the user's hand in the environment, we need to implement the ability to allow both a sampled point model of the data glove and the moving part to participate in the collision response simultaneously. The colliding hand points will affect the haptic collision response in the same manner as the sampled points on the moving part. Furthermore, we can scale the hand model to perform human factor analyses of a task.

A similar needed interaction within a maintenance environment is that of tools, which must fit snugly over certain parts without being repelled, and which must be kinematically constrained to move with fewer degrees of freedom. This ensures that not only can a part be removed, but also that it can be removed using the tool designated for the task.

## References

[1] E. Sanchez, S. Winning, E.S. Boyle. "Automated Support for Maintenance Technical Manuals", Air Force Armstrong Laboratory Technical Paper (AL/HR-TP-1997-0051).

[2] J. Wampler, R. Blue, C.R. Volpe, P. Rondot. "A Virtual Approach to Maintenance Manual Development", Concurrent Engineering - Research and Applications, Proceedings of the Ninth ISPE International Conference on Concurrent Engineering, 27-31 July 2002, Cranfield, United Kingdom, 2002.

[3] United States Air Force, Air Force Research Laboratory Contract Number F33615-01-2-6000.

[4] J. Wampler, R. Blue, C.R. Volpe, P. Rondot. "Service Manual Generation: An Automated Approach to Maintenance Manual Development", Proceedings of the 16th Human Factors in Aviation Maintenance Symposium, April 2-4, 2002, San Francisco, CA., 2002.

[5] J. Wampler, J. Bruno, R. Blue, L. Hoebel, "Integrating Maintainability and Data Development", Proceedings of the Annual Reliability and Maintainability Symposium, January 27-30, 2003, Tamp Bay, FL., 2003.

[6] W.A. McNeely, K.D. Puterbaugh, J.J Troy. "Six Degree of-Freedom Haptic Rendering Using Voxel Sampling." Proceedings ACM SIGGRAPH 99 Conference, August 1999, Los Angeles, CA, pp. 401-408, 1999.

# EVALUATION OF THE PHANToM PLAYBACK CAPABILITY

**Robert L. Williams II** and **Mayank Srivastava**
Department of Mechanical Engineering
Ohio University

**Robert R. Conatser Jr.** and **John N. Howell**
Department of Biomedical Sciences
Ohio University

## ABSTRACT

This article presents implementation and evaluation of a position and force haptic playback system for the PHANToM haptic interface, in the context of our Virtual Haptic Back Project at Ohio University. Playback has the potential to improve virtual palpatory diagnosis training by allowing students to follow and feel an expert's motions prior to performing their own palpatory tasks. No human factors data is presented; rather, this article studies the performance and implementation of our playback system, in terms of how faithful the reproduction of recorded position and force is. We experimentally study the position and force errors upon playback, as a function of our playback parameters: spring stiffness $k$ and zero force radius $r$. Position error decreases with increased $k$ and decreased $r$. However, one cannot increase $k$ or decrease $r$ indefinitely as an unacceptable buzzing effect arises. Force error is not much affected by different $k$ and $r$.

## 1. INTRODUCTION

The Virtual Haptic Back is under development at Ohio University to augment the palpatory training of Osteopathic Medical Students and Physical Therapy and Massage Therapy students (Holland et al., 2002). This project has implemented a high-fidelity graphical and haptic model of the human back on a PC, using the PHANToM interface for haptic feedback.

The current article focuses on the implementation and performance of our PHANToM playback feature, motivated by training needs in the Virtual Haptic Back Project at Ohio University. This article presents position and force error data in PHANToM playback in the Virtual Haptic Back context. This article first presents a brief overview of the Virtual Haptic back, followed by a description of our PHANToM playback capability, and then presentation and discussion of our playback system experiments and results.

## 2. THE VIRTUAL HAPTIC BACK

A virtual back graphics model has been developed, based on measurements taken with a 3D digitizer. Haptic feedback has been programmed, associated with this virtual back model via the PHANToM haptic interface (Fig. 2, Massie and Salisbury, 1994, also http://www.sensable.com/ .

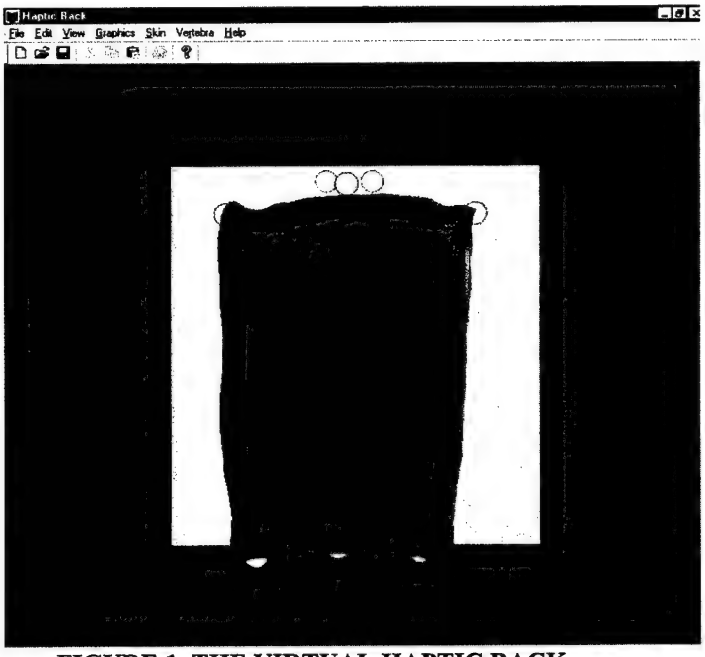



**FIGURE 1. THE VIRTUAL HAPTIC BACK**

The virtual back consists of skin, a spine (C2 at top then C6 through L5), interspinous ligament, scapulae, acromion processes and PSIS (posterior superior iliac spine). In using the Virtual Haptic Back, the student first encounters resistance from compression of the skin and then additional resistances representing underlying bone. The interspinous ligaments joining the spinous processes are palpated as objects with less intrinsic stiffness (more give) than the spinous processes. Transverse processes can also be palpated lateral to the spinous processes and deeper. Each vertebra can rotate in response to pressure applied by the operator to the transverse processes. The resistance to rotation can be set independently for each vertebra. The initial position of each vertebra can also be set independently via menu. The graphics can be set to reveal the underlying bone or not, so that the palpation can be done with or without the aid of seeing the vertebrae on the screen (the real world does not allow this choice!).

## 3. PLAYBACK SYSTEM

In a paper on diagnosing prostate cancer (Burdea et al., 1999), the PHANToM playback mode is used both to analyze a trainee's performance and to show the trainee how an expert approaches prostrate examinations. The same research group is applying general graphics playback in palpation training for detecting subsurface tumors (Dinsmore et al., 1997). In this paper, a data file is written with all inputs from all I/O devices to replay the user's actions graphically. This case does not involve the PHANToM with haptic playback. A second group is using the PHANToM playback feature in their horse ovary palpation simulator (Crossan et al., 2000), to implement a tutor/trainee model.

We have developed a haptic playback system wherein user's position and force interactions with a haptic model may be saved and played back to the PHANToM. This haptic playback has two versions, one in which the recorded interaction forces are played back and the haptic model is turned off and the other in which the recorded forces are not sent but the haptic model is turned on, i.e. we make the PHANToM trace the user's previous path and forces are felt due to the PHANToM interacting with the haptic objects.

To achieve playback in the Virtual Haptic Back, two data files are created during the recording mode. One file records the *XYZ* positions of the PHANToM and the other records its $F_X$, $F_Y$, and $F_Z$ reaction forces. In the original simulation the input comes from the user's hand motions, via the PHANToM encoders. In playback the input is read from the data files; the position playback is achieved as described below.

**Position Playback.** For recording the user's path, points are sampled at a rate of 1000 *Hz*, and saved in a binary text file. These points are read and the PHANToM playback driving force $F$ is calculated using (1). This driving force moves the tip of the PHANToM back through the previously-recorded positions for playback.

$$F = k\left(\|v\| - r\right)\left(\frac{1}{\|v\|}\right)v \qquad (1)$$

In (1), $k$ is the virtual spring constant; $v$ is the vector distance between the current PHANToM position and the next playback point to move to (the center of an attractive spherical force field, see Fig. 2). $r$ is the radius of the spherical region in which there is no force.

The center of the spherical attractive force field is initially located at the PHANToM tip so the PHANToM is within the no-force region. The PHANToM has some play in this spherical no-force region, so $r$ should be small for small position error. The force field is then shifted to the next recorded position. As this is done the PHANToM is moved out of the no-force region. The driving force (1), proportional to the distance $v$–$r$, acts on the PHANToM and attracts it to the zero force region of the shifted force field. The force field is then shifted to the next recorded position and this loop repeats until the end of the playback file is reached.

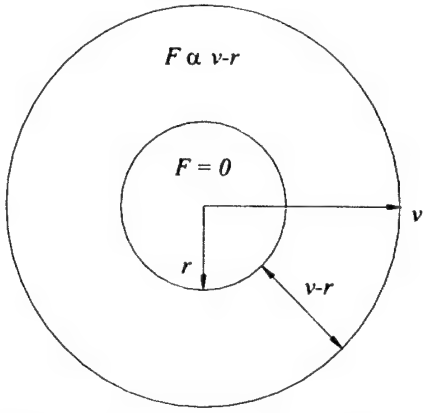



**Figure 2. Spherical Attractive Force Field for Position Playback**

**Force Playback.** During the recording mode all the PHANToM reaction forces are saved to a data file in binary mode. To achieve force feedback during playback, a force field of sufficiently large volume to cover the volume of the haptics-enabled region on the screen is created. The reaction forces are then read and sent to the PHANToM to get the playback force.

## 4. PLAYBACK EVALUATION RESULTS

This section presents the playback experiments and the results obtained. Four different cases were considered: with and without the human finger in the PHANToM thimble and with and without the recorded force with corresponding haptics model disabled and enabled, respectively. We will present here results for without finger case only.

For the experiment an arbitrary path of approximately one minute duration was chosen, with 41,957 path points. The path was made to interact with the virtual human skin, spine, interspinous ligaments, and the scapula. In order to compare the effectiveness of the system under different values of $k$ and $r$, the same path is used for all cases.

The difference between the recorded force and position and those obtained during playback is calculated in the *X*, *Y*, *Z* directions. In this article, a mean square error (*MSE*) measure is used for both force and position:

$$MSE = \frac{\sum_{i=1}^{n}\sqrt{\left(X_{iR} - X_{iP}\right)^2 + \left(Y_{iR} - Y_{iP}\right)^2 + \left(Z_{iR} - Z_{iP}\right)^2}}{n} \qquad (2)$$

$X_{iR}$ is the recorded and $X_{iP}$ the played back $X$ component of position at the $i^{th}$ point; the $Y$ and $Z$ terms are defined in a similar manner. *MSE* for force is defined analogously to (2).

We also calculate the standard deviation to give a measure of the spread of position and force errors over the playback path.

**Position Playback** The results show that smaller $r$ and larger $k$ tend to yield lower position errors. But we cannot reduce $r$ and increase k indefinitely as this introduces a buzzing effect. For the results of Figs. 3, a nominal constant value of $k = 0.38$ *N/mm* was used, and $r$ was varied by steps of 0.10 *mm*. In the results of Figs. 4, a nominal constant value of $r = 0.06$ *mm* was used, and $k$ was varied by steps of 0.02 *N/mm*. In both the plots, standard deviations are included for the case where the playback forces are included (shown in solid blue). Standard deviations are not included for the cases without the recorded forces played back (but with haptics model on, shown in dashed green) simply because the plots are too cluttered. The *MSEs* can be easily compared in the plots.

Using OpenGL, two curves were drawn, the red one representing the recorded positions and the green one showing the path traced by the PHANToM upon playback (see Fig. 5). It was observed that the two lines were close throughout the entire motion, so the position error is constant throughout the trajectory.

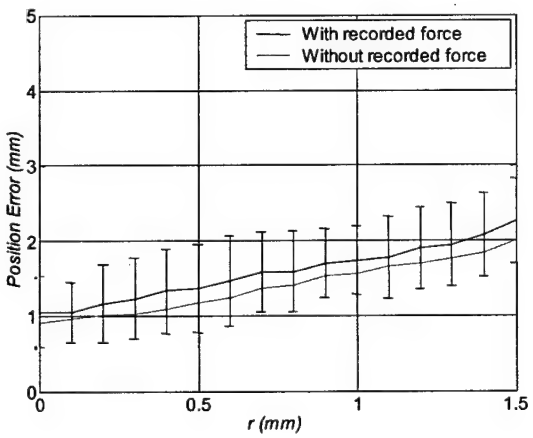



**Figure 3. Position *MSE* vs. *r*, Without Finger**

**Force Playback.** During playback the reaction forces on the PHANToM are compared with the recorded forces and the force mean square error is calculated, similar to the position *MSE* in (2); also, standard deviation is calculated and displayed as in the previous position error plots

The force error obtained is on the order of 0.3 – 0.4 *N* (See Figs. 6 and 7). For the result of Fig. 6, a nominal constant value of $k = 0.38$ *N/mm* was used, while in the results of Fig, 7 a nominal constant value of $r = 0.06$ mm was used.

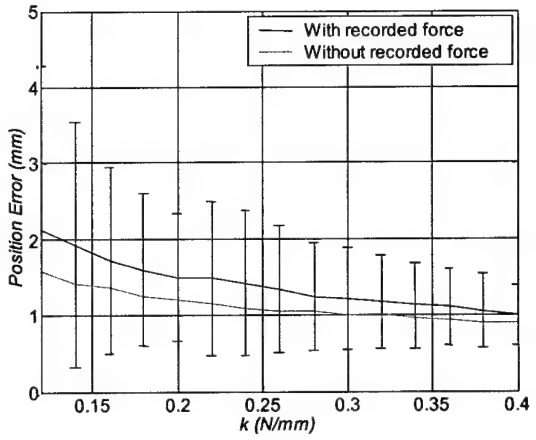



**Figure 4. Position *MSE* vs. *k*, Without Finger**

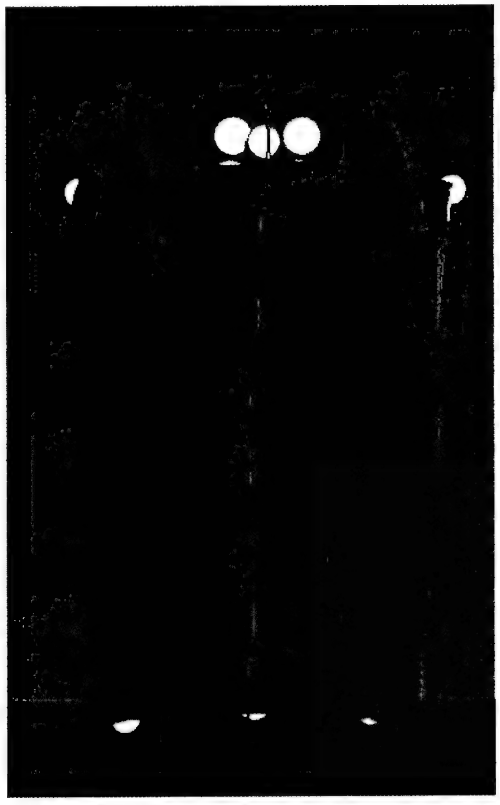


**Figure 5. Playback and Recorded Paths**

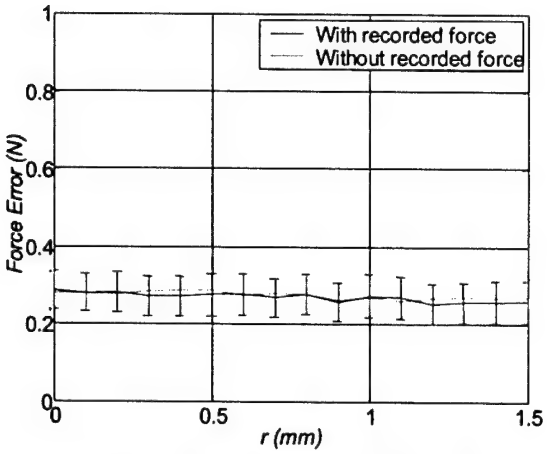



**Figure 6. Force *MSE* vs. *r*, Without Finger**

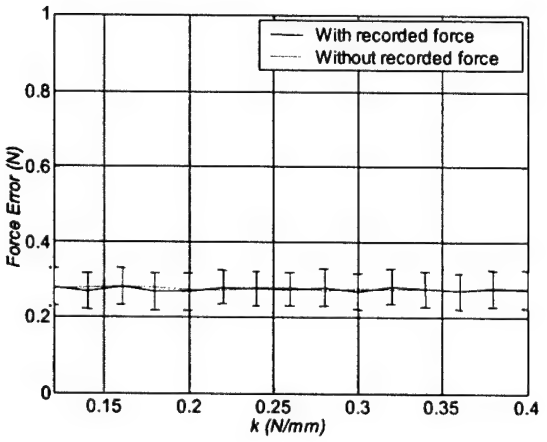



**Figure 7. Force *MSE* vs. *k*, Without Finger**

## 5. Conclusion

The position error increased with increasing *r* and decreased with increasing *k*. The position playback is faithful, observed through numerical, graphical, and subjective means. The force errors remained relatively constant, with little or no dependence on *r* and *k*.

When the system is in normal mode (not playback mode), the PHANToM motors act against the movement of human finger when an obstacle is encountered in the virtual environment. The human finger resists this force feedback to feel the modeled haptic sensations. In playback mode in our system the human finger is passive. Hence, accurate force feedback may not be obtained.

We do record the reaction forces and send them back to the PHANToM. But at the same time we send the driving force for position playback to the PHANToM, to move it through the recorded positions path. These two types of forces (position playback, haptic feedback) interact and hence the desired force feedback may not be achieved. But since the haptic forces are small, the force feedback subjectively appears to be faithful.

## 6. Future Work

A near-term goal in the project is to perform playback experiments with trainees to determine the potential impact of the playback feature on learning palpatory diagnosis tasks using our Virtual Haptic Back. The current article studies only the system performance regarding playback, not including any human trainees. Another future goal is to integrate two PHANToM interfaces, including playback, within the Virtual Haptic Back. Users can then use their thumb and forefinger to palpate as they do in the real world.

## REFERENCES

G. Burdea, G. Patounakis, V. Popescu, and R.E. Weiss, 1999, "Virtual Reality-Based Training for the Diagnosis of Prostate Cancer", IEEE Transactions on Biomedical Engineering, 46(10): 1253-60.

M. Dinsmore, N. Langrana, G. Burdea, and J. Ladeji, 1997, "Virtual Reality Training Simulation for Palpation of Subsurface Tumors", IEEE International Symposium on Virtual Reality and Applications, Albuquerque, NM, March: 54-60.

A. Crossan, S.A. Brewster, S. Reid, and D. Mellor, 2000, "Multimodal Feedback Cues to Aid Veterinary Training Simulations", Proceedings of the First Workshop on Haptic Human-Computer Interaction: 45-49.

K.L. Holland, R.L. Williams, R.R. Conatser Jr., J.N. Howell, and Dennis L. Cade, 2002, "Implementation and Evaluation of a Virtual Haptic Back", Virtual Reality Society

T.H. Massie and K.J. Salisbury, 1994, "*PHANToM Haptic Interface: A Device for Probing Virtual Objects*", ASME International Mech Engr Congress, Chicago, IL, DSC 55(1): 295-299.

# A System for Accurate Collocation of Haptics and Graphics

Karl Reinig, Joshua Eskin, University of Colorado, Center for Human Simulations

I would conservatively estimate that in the last six years I have demonstrated haptic and graphic enabled virtual environments to over a thousand individuals. These individuals range from highly skilled surgeons to first graders. Having witnessed these interactions, I have come to the following conclusion: some people do better with Virtual Reality (VR) than others. Discussions of this variance usually focus on the computer familiarity of the user, which bodes well for the video game player. I would like to suggest the following hypothesis: The major difference in people's VR savvy is their varying ability to adapt to miscues in the virtual environments. If this is true, the following follow:

- The user's performance in a highly accurate virtual environment will not differ significantly from their performance in a real environment.
- There is a significant disparity across users between the correlation of the degradation of skills and degradation of the virtual environment.
- If standards are not developed and maintained, individuals not adept at adapting to miscues will be discriminated against as virtual environments make their way into the mainstream of training and testing.

Many systems have been developed to collocate the apparent graphic position of a virtual scene with a haptic workspace. The use of mirrors for this purpose is particularly popular, as they are inexpensive and have the natural effect of hiding the haptic from the viewer. However, the systems that I have experienced--including the one designed and used at the University of Colorado's Center for Human Simulation--allow for motion of the user's head without appropriately compensating for the implied change in the location and orientation of the virtual camera. The disparity between the true view point and the virtual viewpoint is a direct source of error in the collocation of the haptic and graphic scene.

For mirror systems, two methods come to mind for fixing this disparity: head tracking and fixing the position of the user's head. We have chosen to create a system that does the latter. In our latest system, the user views the virtual environment in much the same way that a person looks through a microscope. This fixes the eye position. There are many other factors that go into making an accurate virtual environment, including:

- calibration of the haptic display device
- calibration of the graphic display device
- graphic refresh rate
- graphic resolution
- proper stereo cues
- proper math

But all of the above are, to a large extent, under the control of the designer.

This presentation introduces a haptic and graphic workstation designed and built with the emphasis on presenting the user with an accurate virtual environment. We expect this system to be a useful display for many of our medical simulators. The more intriguing potential of the system lies in what it may teach us about people using virtual environments.

# An Experimental Study of Performance and Presence in Heterogeneous Haptic Collaboration: Some Preliminary Findings

Margaret McLaughlin, Gaurav Sukhatme, Wei Peng, Weirong Zhu, and Jacob Parks
Integrated Media Systems Center, University of Southern California

**Abstract**

*Based on a distributed architecture for real-time collection and broadcast of haptic information to multiple participants, heterogeneous haptic devices (the PHANToM and the CyberGrasp) were used in an experiment to test the performance accuracy and sense of presence of participants engaged in a task involving mutual touch. In the experiment, the hands of CyberGrasp users were modeled for the computer to which the PHANToM was connected. PHANToM users were requested to touch the virtual hands of CyberGrasp users to transmit randomly ordered letters of the alphabet using a pre-set coding system. Performance accuracy achieved by a small sample of participants was less than optimal in the strict sense: accurate detection of intended location and frequency of touch combined ranged from .27 to .42. However, participants accurately detected the intended location of touch in 92% of the cases. Accuracy may be positively related to pairwise sense of co-presence and negatively related to mean force, force variability, and task completion time.*

## 1. Introduction

In many applications of haptics it will be necessary for users to interact with each other as well as with other objects. We have been working to develop an architecture for haptic collaboration among distributed users. Our focus is on collaboration over a non-dedicated channel (such as an Internet connection) where users experience stochastic, unbounded communication delays [1-5].

In our most recent work, we have focused on improving our system by implementing the synchronization mechanism as a collaboration library. To achieve the desired visual and haptic capabilities for the collaboration program, a model of a human hand was developed for the computer to which the PHANToM was connected. Each segment of the hand is a 3D model. Each component of the hand (the palm, the three segments of each finger, and the two segments of the thumb) was imported separately into the haptic environment. The components were arranged and aligned visually to produce the image of the hand. In the code, the components were organized in a tree-like architecture. To maintain the shape of the hand as it moves during simulation, the movements of components more distal to the wrist are described in the reference frames of adjacent components more proximal to the wrist. For example, the tip of a finger moves with respect to the middle segment of the finger, the middle segment moves with respect to the first segment of the finger, and the first segment moves with respect to the palm. This utilization of relative motion reduces the amount of information required to describe the motion of the hand. Only the rotation angles of each component (relative to its 'parent' component) are needed. The x-y-z coordinates of each component relative to its parent are fixed, and therefore need not be updated. (The only rectangular coordinates that update are for the wrist.) The CyberGrasp computer sends updated transform parameters, which the PHANToM computer uses to generate new coordinate values for the hand.

The other major issue concerning development of the hand model involved contact sensing by the PHANToM. In the haptic environment, regular, solid geometric shapes provide the PHANToM with substantial contact force. This was not so with the hand. Its two-dimensional triangular components would tend to let the PHANToM pass through, with little tactile indication of contact. This problem was solved in a makeshift fashion. A skeleton was constructed out of solid geometric shapes, and placed inside the hand model, to provide contact sensation for the PHANToM. In this solution, PHANToM users see the VRML hand, and feel the skeleton. (We will address this pass-through issue in future work by considering alternative surface representations.)

## 2. Experiment

### 2.1 Experiment Design

Here we report an experiment to evaluate our collaborative system. Some preliminary data were collected from a small-scale study of heterogeneous haptic collaboration, in which one participant, wearing the CyberGrasp, interacts over the Internet with another participant at a remote location (another lab area), who is assigned to a computer with the PHANToM haptic stylus attached. The participants are assigned to

an experimental task that assesses the ability of a subject wearing the CyberGrasp to recognize and discriminate among patterns of passive touch with respect to frequency and location, and the ability of the PHANToM user to communicate information effectively through active touch.

We can call the participants, respectively, P1 and P2. Participant P1's monitor shows a model of P2's (the CyberGrasp wearer's) hand, which is updated in real time to reflect the current location and orientation of P2's hand and fingers. P1's task is to use a PHANToM haptic stylus, the tip of which is represented by an enlarged round ball, to communicate information to P2 through a tactile "Morse code" which maps the number of times a particular digit is "touched" onto the 26 letters of the alphabet. A keypad showing this mapping is visible on P1's display. P2 holds her hand stationary during this process. P2 then enters (or has entered for her) the letters she thinks she has received into a keypad visible on her own display, from which they are logged into a word file. On P2's monitor, only the keypad is visible; P2 is unable to see the hand display. The two subjects can communicate with each other via a text-based messaging system, sending pre-programmed messages such as "Experiment begins," "New word," "New letter," "Ready for next letter," "Next trial," and so on. All text messaging between the partners is logged and time-stamped.

### 2.2 Participants

Participants were recruited through notices placed on bulletin boards and circulated through the campus BBS and student mailing lists. For participants to be eligible they had to be right-handed, over 18, and unacquainted with any other person planning to volunteer for the study. Upon acceptance into the participant pool, volunteers were randomly assigned to the "CyberGrasp" condition or the "PHANToM" condition and directed to one of two lab locations on campus. The target N of pairs for the study is 25; we are currently collecting data and report here on a very small sample of three pairs.

### 2.3 Procedure

Upon arrival at their respective lab locations, participants were then provided with a PowerPoint tutorial tailored to the type of haptic device to which they were assigned. The tutorial introduced them to haptics and in the case of the PHANToM user provided an opportunity for them to practice using the device for active exploration of a digital object. The CyberGrasp user's role in the experiment was to be a passive recipient of touch and as such did not "practice" but was simply taken through the calibration process upon completion of the tutorial.

The experimenters in their respective lab locations communicated with Motorola Talkabout radios to coordinate opening of the collaborative workspace. Participants joined in the workspace over an ordinary Internet connection. Next, they were provided with oral instructions which reinforced the explanations of the experimental task provided in the tutorials. As an added check to ensure that any obtained errors in location or frequency of touch were attributable only to the haptic interaction and not to participants' misreading of the code, participants were required to state aloud which finger they intended to touch and how many times they intended to touch it. When the oral instructions were completed, the investigator supervising the PHANToM user started a process for recording the haptic data stream (capturing and time-stamping applied force at 10ms intervals) [6] and signaled through the keypad that the experiment was to begin. The investigator supervising the CyberGrasp user minimized the collaborative workspace window so that the participant would respond only to the haptically communicated information. The pair of participants worked their way through a word list that contained all 26 letters of the alphabet, sorted randomly into nine sets of three- and two-letter words. At the conclusion of the trials, participants completed a post-task evaluation with items adapted from the Basdogan et al. measure of co-presence. [7] Among the questions of interest: Did the subject feel present in the haptic environment? Did the subject feel co-present with the remote partner? Did the subject believe that the remote partner was a real person? Subjects were also asked to make attributions about their partners with respect to standard dimensions of perception (unsocial-social; sensitive-insensitive; impersonal-personal; cold-warm) (Sallnas, Rassmus-Grohn, & Sjostrom, 2001). [8] (Findings with respect to the person perception data will be reported elsewhere.)

## 3. Results

### 3.1 Accuracy

Performance for all three pairs was seemingly poor. Accuracy of the best pair was 42.3%. The next best pair got 38.4% of the letters correct, and the least effective pair only 26.9%. However, examination of the confusions data indicates that, with the exception of the thumb (see Table 1, below), most of the confusions

resulted from an overestimate of the number of times the finger was tapped. Only 8% of the errors made were egregious, e.g., the recipient was confused about which finger was being tapped. We present two of the confusions matrices below, the matrix for the ring finger (Table 1) and for the thumb (Table 2).

Table 1. Table of Confusion for the ring finger *

| | | Reported Number of Taps | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| Number of Taps Called for by Code | 1 | | x | | x | | |
| | 2 | | | x | | | |
| | 3 | | | | xx x | | |
| | 4 | | | xx | | | |
| | 5 | | | | | | |
| | 6 | | | | | | |

*Confusions as to number of taps are represented by small x's.

Table 2. Table of Confusion for the thumb*

| | | Reported Number of Taps | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | Finger Mid. | Pinky |
| Number of Taps Called for by Code | 1 | | x | | | | | | **X** |
| | 2 | x | | | | | | | |
| | 3 | x | | | | | x | | |
| | 4 | | x | | | | | **X** | |
| | 5 | xx | | | | | x | | |
| | 6 | | | | | xx | | | |

*Confusions as to number of taps are represented by small x's. Confusion as to location of touch are represented in bold capital X's at the right-hand side of the table

3.2 Performance and presence variables

Performance values for the participant pairs for (a) the number of points at which there was measurable force, (b) minimum force, (c) maximum force, (d) force variability (standard deviation of sampled values), (e) time to task completion, and (f) accuracy (percentage of correctly decoded letters) are presented in Table 3 (a)(b). Co-presence data is also reported in Table 3 (b). Comparison of the most accurate pair (Pair 3) and least accurate pair (Pair 1) indicates that the less accurate pair had far fewer sampled points with measurable force, greater mean force, higher force variability, and longer time to task completion (although taken over all pairs these relationships are not monotonic with respect to task completion time). Comparison of the most and least accurate pairs (Pair 3, Pair 1) indicated that perceived co-presence was rated higher by the more accurate subjects, and by both members of the pair. However, taken over all pairs the relationship between co-presence and accuracy is not monotonic for the PHANToM users.

Table 3. Performance variables and co-presence ratings for participant pairs

(a)

| Pair | N Points Measurable Force | Min Force | Max Force | Mean Force | Force S.D. |
|---|---|---|---|---|---|
| 1 | 0494 | .0005 | .8530 | .2788 | .2623 |
| 2 | 6607 | .0005 | .8974 | .2278 | .2501 |
| 3 | 3688 | .0005 | .9039 | .1240 | .1699 |

(b)

| Pair | N Points Measurable Force | Min Force | Max Force | Mean Force | Force S.D. |
|---|---|---|---|---|---|
| 1 | 0494 | .0005 | .8530 | .2788 | .2623 |
| 2 | 6607 | .0005 | .8974 | .2278 | .2501 |
| 3 | 3688 | .0005 | .9039 | .1240 | .1699 |

**4. Discussion**

Although the accuracy levels obtained in this small sample appeared to be low, it was in fact the case that in 92% of the instances the passive touch recipient (the CyberGrasp wearer) was able to distinguish which digit was being touched. The bulk of the obtained errors resulted from an overestimate by one of the number of times the digit was tapped. There are a number of possible explanations, two of which seem most likely. First, there may have been incidental vibration of the CyberGrasp unrelated to the intended activity of the PHANToM user. If that were the case, however, there would have been more reports by the

CyberGrasp user that his or her partner was "trying to touch several fingers" at once. Our experience to date indicates that touch on some of the middle digits can produce vibration in neighboring digits. What we did observe is that the PHANToM user, having only a single point of contact, often "missed" connecting with the partner's fingers on the first several tries at a new letter, and would frequently follow up on a light application of force to the finger with a stronger one, even though the deformation of the finger model on the first occasion of contact was clearly visible on the monitor. Further practice and more intensive coaching in the tutorial on interpreting the visual indicators of contact should improve results in future experiments. Incorporation of immediate feedback about performance accuracy on initial trials should also improve performance on subsequent trials.

Poorest performance was obtained for information communicated through taps to the thumb. In addition to the two cases in which the CyberGrasp wearer perceived a tap intended for the thumb to be directed to another digit, the thumb confusions matrix shows clearly that the passive recipient consistently underestimated the number of taps relative to the number intended by the active partner. The most plausible explanation has to do with the comparative difficulty given the orientation of the hand of contacting the palmar face of the thumb with the PHANToM. Many of the taps were applied to the dorsal side.

Although we expected that greater mean force would be associated with greater accuracy; that does not appear to have been the case in this small sample. There may be a force threshold which once met is adequate for detection, or it may be that the necessary level of force varies with receivers. And it may be that mean force over multiple trials is not a meaningful measure. We hope to sort out these issues as we collect additional data. Our expectations with respect to number of points of measurable force, force variability and task completion time are so far being confirmed with the data at hand; more variable application of force, less frequent application of force, and longer time to complete the task appear to have a negative impact on pairwise accuracy.

Finally, there is some very slight evidence that sense of co-presence may turn out to be stronger in more accurate pairs, although we are far more comfortable looking for trends in performance over multiple trials with a handful of participants than we are in looking for relationships between overall performance accuracy and subjective, retrospective assessments of the collaborative environment given only the three pairs. We expect to find additional evidence in support of this relationship as we continue to collect data.

**References:**


[1] Hespanha, J., McLaughlin, M. L., & Sukhatme, G. (2002). Haptic collaboration over the Internet. In McLaughlin, M. L., Hespanha, J., & Sukhatme, G. (Eds.). *Touch in Virtual Environments: Haptics and the Design of Interactive Systems*. Prentice Hall.

[2] Hespanha, J., Sukhatme, G., McLaughlin, M., Akbarian, M., Garg, R., & Zhu, W. (2000). Hetereogeneous haptic collaboration over the Internet. *Proceedings of the Fifth Phantom Users' Group Workshop*, Aspen, CO.

[3] McLaughlin, M. L., Sukhatme, G., Hespanha, J., Shahabi, C., Ortega, A., & Medioni, G. (2000). The haptic museum. *Proceedings of the EVA 2000 Conference on Electronic Imaging and the Visual Arts*. Florence, Italy.

[4] Shahabi, C., Ghoting, A., Kaghazian, L., McLaughlin, M., & Shanbhag, G. (2001). Analysis of haptic data for sign language recognition. *Proc. First International Conference on Universal Access in Human-Computer Interaction (UAHCI)*, New Orleans, Louisiana, 5-10 August 2001. Hillsdale, NJ: Lawrence Erlbaum.

[5] Sukhatme, G., Hespanha, J., McLaughlin, M., & Shahabi, C. (2000). Touch in immersive environments. *Proceedings of the EVA 2000 Conference on Electronic Imaging and the Visual Arts*, Edinburgh, Scotland.

[6] Shahabi, C., Kolahdouzan, M., Barish, G., Zimmermann, R., Yao, D., & Fu, L. (2001, June). Alternative techniques for the efficient acquisition of haptic data, ACM SIGMETRICS/Performance 2001, Cambridge, MA.

[7] Basdogan, C., Ho, C., Srinivasan, M. A., & Slater, M. (2000) An experimental study on the role of touch in shared virtual environments. ***ACM Transactions on Computer Human Interaction***, *7*(4), 443-460.

[8] Sallnas, E. L., Rassmus-Grohn, K. & Sjostrom, C. (2001). Supporting presence in collaborative environments by haptic force feedback. ***ACM Transactions on Computer-Human Interaction***, *7* (4), 461-476.

# Comparison of Human Haptic Size Discrimination Performance in Real and Simulated Environments

*Marcia Kilchenman O'Malley*
*Mechanical Engineering and Materials Science*
*Rice University*
*Houston, Texas*
*omalleym@rice.edu*

## Abstract

*The performance levels of human subjects in size discrimination experiments in both real and virtual environments are presented. The virtual environments are displayed with a Phantom desktop three degree-of-freedom haptic interface. Results indicate that performance of the size discrimination tasks in the virtual environment is comparable to that in the real environment, implying that the haptic device does a good job of simulating reality for these tasks. Additionally, performance in the virtual environment was measured at below maximum machine performance levels for two machine parameters. The tabulated scores for the perception tasks in a sub-optimal virtual environment were found to be comparable to that in the real environment, supporting previous claims that haptic interface hardware may be able to convey, for these perceptual tasks, sufficient perceptual information to the user with relatively low levels of machine quality in terms of the following parameters: maximum endpoint force and maximum virtual surface stiffness. Results are comparable to those found for similar experiments conducted with other haptic interface hardware, further supporting this claim.*

## 1 Introduction

This paper presents a comparison of human haptic performance in real and virtual environments. Results support the case that haptic interfaces are good at simulating real objects, and indicate that they can do so without excessive machine performance demands for the tasks described here. Comparisons of performance in real and virtual environments have been made in the past. Typically these comparisons are made with the virtual environment display operating such that the best achievable representation of reality is presented to the user. For example, completion times for a pick and place task performed in a real-world control environment and in three virtual conditions were presented by Richard et al. [1]. Their findings showed that for the pick and place task, completion times, a measure of task performance, were lower for the real-world control environment than for each of the three virtual environments tested. However, accuracy for depth and lateral placement were comparable for one haptic display and the real-world control. Similarly, Buttolo et al. [2] used comparative methods to study the differences in performance of simple manipulation tasks with real objects, with a virtual reality simulation containing force feedback, and remotely with a master and slave system, also with force feedback. Their findings also showed that performance with the virtual environment was similar to that with real objects. This paper takes a similar comparative approach to verify the quality of a haptic device in simulating realistic virtual environments for a simple perceptual task of size discrimination, where subjects determine which of two objects placed side by side is larger. Additionally, the quality of the haptic device, in terms of two parameters, is degraded and another performance comparison is made.

## 2 Methods

### 2.1 Virtual Environment Apparatus

The Phantom desktop was used to simulate the virtual environments. Hardware specifications are listed in Table 1.

**Table 1. Phantom Desktop hardware specifications**

| | |
|---|---|
| Workspace | 16x13x13 cm |
| Maximum force | 6.4 N |
| Maximum continuous force | 1.7 N |
| Force feedback | 3 DOF |
| Position sensing | 6 DOF |

### 2.2 Testing Environments

In both the real and virtual environments, subjects held a stylus with the dominant hand and entered responses on a computer keyboard with the non-dominant hand. The dominant hand was shielded from view with a curtain for both tests. A frame for the curtain was constructed from a cardboard box. The face closest to the subject was cut out and replaced with a curtain to

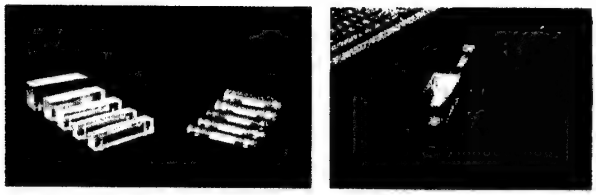


**Figure 1. Photograph of the real blocks and the environment for a square ridge size discrimination task.**

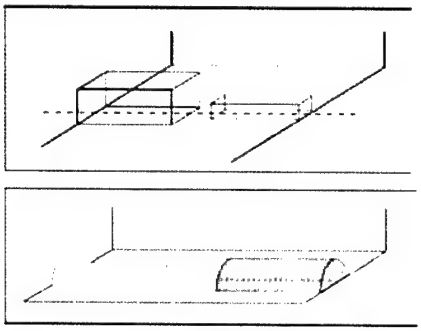


**Figure 2. 3-D model of the simulated environment for the square and round ridge size discrimination tasks**

**Figure 3. Test subject seated at testing station for virtual environment experiments. (The box and curtain used to obstruct the subject's view are removed in this picture)**

allow free movement of the hand. The back of the box was also removed to allow the test administrator to change blocks for the real environment. The box was secured to the desktop throughout the practice and testing sessions.

### 2.2.1 Real Environment

The real environment consisted of square and round cross-section blocks, machined out of acrylic, mounted on pegs on an aluminum plate. The blocks were covered with clear contact paper to account for any texture irregularities due to machining. The subject used an aluminum stylus to explore the environment. These blocks and the test plate are shown in Figure 1.

### 2.2.2 Virtual Environment

Square and round cross-section blocks were displayed haptically with the Phantom Desktop. Figure 2 shows a visual representation of the virtual environment. Subjects explored the environment with the Phantom stylus as shown in Figure 3.

### 2.3 Experimental Paradigms

Perception experiments were conducted for ridges of square and semicircular cross-sections, and were conducted in a real environment and several virtual environments (high fidelity, low fidelity-force, and low fidelity-stiffness). The order of testing was varied for each subject to ensure that learning effects were not a factor. Short practice sessions (10 trials each) were conducted prior to each experiment.

### 2.4 Subjects

Ten test subjects were used for the size discrimination experiments. A cross-section of subject types (gender, dominant handedness, and experience with haptic devices) was chosen for each of these experiments.

### 2.5 Procedures

In each experiment, the subject was asked to feel the exterior of the two ridges and determine which was larger, entering their response on the keyboard (left or right). One of the two ridges was always the base size, with an edge length of 20 mm. The second ridge had an edge length of 20, 22.5, 25, or 30 mm. In both the real and virtual environment experiments, twenty trials of each stimulus pair were presented to the subject. In all, subjects sat for eight test sessions of eighty trials which were each preceded by a ten trial practice session.

### 2.5.1 Machine Parameters

In order to create low-fidelity environments, two machine parameters were selected to describe haptic interface machine performance, namely maximum force output and time delay. Force output correlates to torque output limits of motors, and increased torque output requirements are typically proportional to motor cost and size. Time delays are unfavorable in a real-time system, and reduction of time delay usually requires faster computing speed and higher quality electronics, each coupled to an increase in price. These two quantifiable machine parameters are easily understood by designers and are typical measures of system quality. During experimentation in the low-fidelity virtual environments, these machine parameters were lowered to minimum levels for good performance of the size discrimination task as determined by O'Malley and Goldfarb [3, 4]. To limit the force output of the manipulator, the output command force was saturated at 4 N for each trial [3]. This was accomplished by creating new classes in GhostSDK called WeakCube and WeakCylinder that take the maximum output force as an input. These classes were

based upon the GstCube and GstCylinder classes in GhostSDK. For the stiffness low fidelity simulations, $k$ was set to 450 N/m and $b$ to 45 Ns/m as recommended in [4].

### 2.5.2 Experiments

Experiment 1 – Real

Testing was conducted with the acrylic blocks and aluminum stylus for both square (A) and round (B) cross-section ridges.

Experiment 2 – High-Fidelity Virtual

Testing was conducted with the Phantom Desktop at default values for force and stiffness. Tests were conducted for both square (A) and round (B) cross-section ridges.

Experiment 3 – Low-Fidelity Virtual: Force

Testing was conducted with the Phantom Desktop at the default value for stiffness and a maximum output force of 4 N. Tests were conducted for both square (A) and round (B) cross-section ridges.

Experiment 4 – Low-Fidelity Virtual: Stiffness

Testing was conducted with the Phantom Desktop at the default value for force and a virtual surface stiffness of 470 N/m. Tests were conducted for both square (A) and round (B) cross-section ridges.

## 3 Results

Results for all experiments are presented in Figures 4 (square cross-section ridges, all environments) and 5 (round cross-section ridges, all environments). Results are shown as percent correct scores and are the average results across all test subjects. Standard errors are shown with error bars.

## 4 Discussion

In Figure 4, we see that performance in the high fidelity virtual environment is comparable to that in both low-fidelity virtual environments for size discrimination of ridges with square cross-section. At a size difference of 1.25 mm, performance is best in the real environment, although this result is not significant as determined by an analysis of variance (ANOVA). At all other sizes, performance in the real environment is comparable to that in the high and low fidelity environments.

Figure 5 shows results for all round cross-section size discrimination experiments. Again, we see comparable performance between the high and low fidelity virtual environments at all size differences. At the 1.25 and 2.5 mm size differences, performance appears to be slightly better in the real environment,

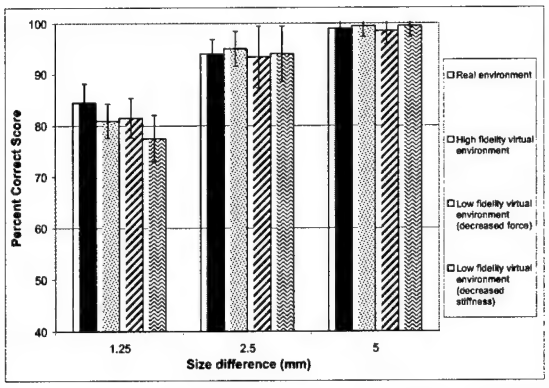



**Figure 4. Results for all size discrimination experiments with square cross-section ridges (1A, 2A, 3A, and 4A)**

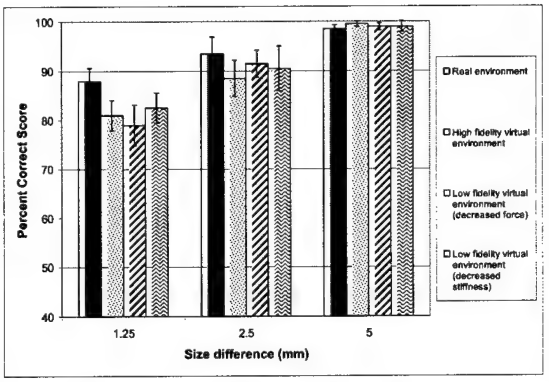



**Figure 5. Results for all size discrimination experiments with round cross-section ridges (1B, 2B, 3B, and 4B)**

although this result is not significant by the ANOVA. Overall, the two-way ANOVAs showed that there were no significant differences in performance when comparing based on environment (real, high fidelity virtual, low fidelity virtual-force, or low fidelity virtual-stiffness).

One subject commented that the low fidelity didn't feel much different than the high-fidelity. This comment supports the author's claim that low fidelity environments may be sufficient for some perceptual tasks.

## 5 Conclusions

The findings of these experiments, in which performance in a real environment was compared to performance in an environment displayed with a Phantom desktop for a size discrimination perception task, indicate that the Phantom does a fairly good job of approximating reality for the block environments described here. Not only do these results support the case that haptic interfaces are good at simulating real environments for these perceptual tasks, but they also

show that they can do so without excessive machine performance demands. Results are similar to those found for other haptic interface hardware.

## 6 Acknowledgements

The work presented herein was supported by NASA Grant NGT-8-52859. The author wishes to thank the volunteers who served as test subjects and the following students who contributed to this work: Andrea Lubawy, Matt Cuddihy, Tony Kellems, and Shannon Hughes.

## 7 References

[1] P. Richard and Ph. Coiffet, "Dextrous Haptic Interaction in Virtual Environmnets: Human Performance Evaluations," Proceedings of the IEEE International Workshop on Robot and Human Interaction, pp. 315-320, 1999.

[2] P. Buttolo, D. Kung, and B. Hannaford, "Manipulation in Real, Virtual, and Remote Environments." Intelligent Systems for the 21st Century. Systems, Man, and Cybernetics, Vol. 5, pp. 4656-4661, 1995.

[3] M. O'Malley and M. Goldfarb, "The Effect of Force Saturation on the Haptic Perception of Detail." IEEE/ASME Transactions on Mechatronics, Vol. 7(3), pp. 280-288, 2002.

[4] M. O'Malley and M. Goldfarb, "The Implications of Surface Stiffness for Size Identification and Perceived Surface Hardness in Haptic Interfaces," Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1255-1260, 2002

will provide a solid foundation for the establishment of the HELS method as a potentially viable noise diagnostic tool. [Work sponsored by NSF and Ford Motor Company.]

**11:00**

**2aSA7. Determination of the optimal number of expansion terms in the HELS method.** Byoung-Duk Lim (Dept. of Mech. Eng., Yeungnam Univ., 214-1 Dae-dong, Gyongsan, Kyungbuk, 712-749, Korea) and Sean F. Wu (Wayne State Univ., Detroit, MI 48202)

The Helmholtz equation least-square (HELS) method has been proven to be an efficient way of reconstructing acoustic radiation from a vibrating structure [Wu, J. Acoust. Soc. Am. **107**, 2511–2522 (2000)]. In this method, the optimal number of expansion terms is determined by a heuristic method of searching the minimum–norm errors of the reconstructed acoustic pressures with respect to the measured data. While this approach works, it is time consuming. In this paper, a new method of *a priori* determination of the optimal number of expansions is developed based on the eigenvalue analysis of the measured information. Assuming a Gaussian white measurement noise, the hypothesis of equal noise eigenvalues is tested based on the statistical properties of the eigenvalues. This hypothesis test yields an upper bound of the optimal number of expansions in general. Also developed is a method for testing the correlation coefficients after eliminating some principal components, which gives a lower bound of the optimum number of expansions. Since the curve of minimum–norm errors in the previous work shows local minima when the number of terms is squares of the integer, the range produced by the present method leads to only a few candidates for the optimal number of expansions. Examples of reconstructing the radiated acoustic pressure field from a vehicle front end using the optimal number of expansion terms thus obtained are demonstrated in parallel with the results of the *a posteriori* method discussed by Wu. [Work supported by NSF.]

**11:15**

**2aSA8. Vibro-acoustography of small spheres.** Shigao Chen, Mostafa Fatemi, and James F. Greenleaf (Dept. of Physiol. and Biophys., Mayo Clinic, Rochester, MN 55905, chen.shigao@mayo.edu)

Vibro-acoustography is a method for imaging the acoustic energy emitted from objects in response to an oscillatory radiation force produced by two interfering focused beams of ultrasound [M. Fatemi and J. F. Greenleaf, Science **280**, 82–85 (1998)]. To facilitate quantitative study, a model is presented that describes the behavior of an elastic sphere in two plane ultrasound waves of slightly different frequencies. The vibrating velocity and the resulting acoustic emission of the sphere at the difference frequency are derived from the radiation force and the impedance of the sphere. For small steel spheres in water, the velocity and emission are predicted to have a simple relationship with the difference frequency and the size of the sphere. Experiments on small stainless steel spheres of different size are carried out for a number of difference frequencies, with a laser interferometer to measure the velocity of the sphere and a calibrated hydrophone to measure the acoustic emission from the sphere. The experimental results fit the model description well. This means that vibro-acoustography may be useful in quantitative measurement.

**11:30**

**2aSA9. Backscattering enhancements associated with antisymmetric Lamb waves on circular plates in water: Observations and imaging of the plate surface using acoustic holography.** Brian T. Hefner and Philip L. Marston (Phys. Dept., Washington State Univ., Pullman, WA 99164, bhefner@mail.wsu.edu)

When a tilted circular plate is illuminated with high-frequency sound in water, significant enhancements occur when the angle of incidence corresponds to a Lamb wave coupling angle. Previously, scattering associated with the lowest-order symmetric Lamb wave, $s_0$, was studied using acoustic holography and modeled using quantitative ray methods [B. T. Hefner and P. L. Marston, J. Acoust. Soc. Am. **107**, 2847 (2000)]. The present investigation focuses on backscattering associated with the antisymmetric Lamb waves. Above the cutoff frequency of the first-order antisymmetric wave, $a_1$, enhancements are observed which are associated with both the $a_1$ wave and the $a_0$ wave. For these enhancements, however, mode conversion between these modes plays a significant role. To examine the mode conversion and to identify the scattering mechanisms, acoustic holography was used to image the plate surface when the plate is insonified at the $a_1$ wave coupling angle. From the imaged wave field, the enhancement at this angle involves a wave which moves around the circumference of the plate: a "whispering gallery" mode involving the $a_1$ wave or a flexural edge wave. Both of these possibilities are considered as well as a possible scattering mechanism at the $a_0$ wave coupling angle. [Work supported by the Office of Naval Research.]

**11:45**

**2aSA10. Diagnostics of vibration and noise via Fourier transforms: A new approach.** Leonid M. Gelman (Dept. of Nondestructive Testing, Natl. Tech. Univ. of Ukraine, 37, Peremogy Pr., Kiev, 252056, Ukraine), Ivan V. Petrunin (Natl. Tech. Univ. of Ukraine, Kiev, 252056, Ukraine), and Vladimir T. Shirkov ("Motor-Sich" Co., Zaporozh'e, 01202, Ukraine)

A new general optimal approach and new diagnostic features are proposed, for those cases where one- and multidimensional Fourier transforms are used for diagnostics of vibration and noise. For the forced oscillation vibroacoustical diagnostics method with Gaussian excitation, the optimal nonlinear transformation of the proposed features is received. It was shown (1) The power spectral density is optimal transformation for considered diagnostics only for specific situations; (2) The phase spectrum is not optimal transformation for considered diagnostics; and (3) The proposed approach provides the increment of diagnostics effectiveness in contrast with usage of power spectral density. This is in contrast to most applications concerning diagnostics of vibration and noise, where power spectral density is used.

2a TUE. AM

# Remote measurement of material properties from radiation force induced vibration of an embedded sphere*

Shigao Chen, Mostafa Fatemi, and James F. Greenleaf

Basic Ultrasound Research Laboratory, Department of Physiology and Biophysics

Mayo Clinic, Rochester, MN 55905

Suggested running title: Vibration of a Sphere

Corresponding author: James F. Greenleaf, Basic Ultrasound Research Laboratory, Department of Physiology and Biophysics, Mayo Clinic, Rochester, MN 55905 USA

Tel: 507 284-8496, Fax: 507 266-0361, email: jfg@mayo.edu

*Part of this work was presented in "Vibro-acoustography of small spheres" at 140th Meeting of Acoustical Society of America, Newport Beach, CA, December 2000.

*Abstract*—A quantitative model is presented for a sphere vibrated by two ultrasound beams of frequency $\omega_1$ and $\omega_2$. Due to the interference of two sound beams, the radiation force has a dynamic component of frequency $\omega_2 - \omega_1$. The radiation impedance and mechanical impedance of the sphere are then used to compute the vibration speed of the sphere. Vibration speed versus vibration frequency is measured by laser vibrometer on several spheres, both in water and in gel phantom. These experimental results are used to verify the model. This method can be used to estimate the material properties of the medium (e.g., shear modulus) surrounding the sphere.

PACS numbers: 43.20.Tb, 43.25.Qp, and 43.30.Jx

## I. INTRODUCTION

The study of objects in terms of their mechanical response to external forces is of considerable interest in material science and medical diagnosis. Changes of elasticity of soft tissues are often related to pathology. The diagnostic value of a characteristic of an object depends upon the range of variation of that characteristic as a function of the state of the object. For soft tissues, the range of shear modulus can vary over three orders of magnitude [Sarvazyan *et al.*, 1998]. Thus imaging techniques that exploit the shear modulus, or stiffness, of an object have great potential in medical application. Elasticity imaging [Gao *et al.*, 1996], a subject extensively investigated in recent years, is a quantitative method that measures the mechanical properties of tissue. The general approach is to measure the response of tissue to an excitation force and use it to reconstruct the elastic parameters of the tissue. These parameters are usually related to the shear modulus, or "hardness" of the tissues being imaged.

Vibro-acoustography [Fatemi *et al.*, 1999] is new imaging modality, which can image the "hardness" of an object. Figure 1 is a brief diagram of a vibro-acoustography imaging system. The confocal transducer has a center disk and an outer ring that introduce two ultrasound beams to the same focal spot. These two ultrasound beams have slightly different frequencies: for example, $\omega_1$=1.001 MHz, and $\omega_2$=0.999 MHz. At the focal spot, the interference of these two beams will cause the object to vibrate at the beat frequency, in this case, at $\Delta\omega$=2 kHz. The resulting acoustic emission contains information about the local material properties of the object and can be detected by the

acoustic hydrophone. Scanning the focal point of the transducer in a raster manner will generate a 2D image of the object.

Vibro-acoustography is particularly useful for imaging hard inclusions in soft material. For example, it has been used to image calcification in human arteries [Fatemi *et al.*, 1998], fractures in metal heart valves [Fatemi *et al.*, July 2000], and microcalcifications in breast tissue [Fatemi *et al.*, 2002]. Other than imaging, another potential application of this technique is material characterization. In this paper, we explore this possibility. We speculate that by evaluating the response of the object to many excitation frequencies, we can estimate the mechanical properties, such as the complex shear modulus, of the object. A model is proposed for a spherical target embedded in a viscoelastic homogenous medium. The goal is to estimate the mechanical properties of the medium by evaluating the displacement frequency response of a embedded spherical target. In this approach we excite the target (e.g., a sphere) at a fix position, and monitor its response as we change the excitation frequency ($\Delta\omega$). By fitting the model to the data, material properties of the medium can be measured.

One potential medical application of this method is to estimate the mechanical properties of breast tissue, using microcalcification in the breast as a target. Breast microcalcifications are commonly found within both benign and malignant lesions [Fatemi *et al.*, 2002]. Previous studies suggest that malignant breast tissues are considerably stiffer than benign breast tissues [Krouskop *et al.*, 1998]. Therefore, information about tissue properties obtained from the vibration of microcalcification has diagnostic value. Other potential applications of this method can be nondestructive evaluation for materials such as polymers or other soft materials. The nature of this

technique allows us to remotely access the mechanical properties of the medium around a target.

The focus of this paper is to develop a quantitative model for a sphere vibrated by two ultrasound beams of different frequency in a homogeneous medium, and to design experiments to verify the model. The paper is organized as follows. The dynamic radiation force used to vibrate the sphere is derived in Section II. Section III gives the radiation impedance of an oscillating sphere, from which the vibrating velocity of the sphere is calculated. The theory and experimental results are compared in Section IV. Section V discusses some practical application issues of this research. The last Section is the summary.

## II. DYNAMIC RADIATION FORCE ON A SPHERE

The acoustic radiation force is a time-average force exerted by an acoustic field on an object. This force is caused by a change in the energy density of an incident acoustic field. Thus, an object in the wave path that absorbs or reflects sound energy is subjected to the acoustic radiation force. This force is usually a steady force, given that the intensity of the incident sound field does not change over time. In vibro-acoustography, the incident ultrasound is modulated such that its energy density changes sinusoidally at a low frequency (in the order of a few kilohertz). Thus, the radiation force on the object is oscillatory. For a general description of this dynamic radiation force on an arbitrary object, see Fatemi and Greenleaf (1999). In the following paragraphs, we will focus only on issues relevant to the application treated in this paper.

Consider a plane ultrasound wave incident on a sphere immersed in fluid. The radiation force on the sphere is along the incident wave direction and its magnitude is

$$\langle F \rangle = \pi a^2 Y \langle E \rangle, \tag{1}$$

where $\pi a^2$ is the projected area of the sphere, Y is the radiation force function, and $\langle E \rangle$ is the time-averaged energy density of the ultrasound [Hasegawa *et al.*, 1969]. The symbol $\langle\ \rangle$ represents time average. The coefficient Y, which can be determined from the material properties of the sphere and the fluid medium, is usually expressed as a function of $k \cdot a$, where k is the wave number of the incident wave and a is the sphere's radius. Consider another situation where the incident pressure field p(t) consists of two sine waves of slightly different frequency

$$p(t) = P_0 Cos\omega_1 t + P_0 Cos\omega_2 t = 2P_0 Cos\left(\frac{\omega_1 - \omega_2}{2} t\right) Cos\left(\frac{\omega_1 + \omega_2}{2} t\right), \tag{2}$$

which is equivalent to an amplitude-modulated sine wave. We have not included position as an argument in Eq. (2), since it is not relevant to the illustration here. For typical applications in this paper, $(\omega_1 - \omega_2)/2$ is at least three orders of magnitude smaller than $(\omega_1 + \omega_2)/2$. Thus, $k_1 a \approx k_2 a$, leading to $Y(k_1 a) \approx Y(k_2 a)$, which we can denote as $\overline{Y}$. The time-averaged energy density of this field is [Fatemi *et al.*, 1999]

$$\langle E \rangle = \left[2P_0 Cos\left(\frac{\omega_1 - \omega_2}{2} t\right)\right]^2 / \rho c^2 = [1 + Cos(\omega_1 - \omega_2)t]\frac{2P_0^2}{\rho c^2}, \tag{3}$$

where $\rho$ and c are the density and sound speed of the fluid. Therefore the radiation force on the sphere has a dynamic component of frequency $\omega_1 - \omega_2$

$$\langle F_d \rangle = \pi a^2 \overline{Y} \langle E_0 \rangle Cos(\omega_1 - \omega_2)t\,, \tag{4}$$

where $\langle E_0 \rangle = 2P_0^2 / \rho c^2$.

In practice, the incident beams from the confocal transducer may differ from ideal plane waves, and their intensity may not be equal. The medium surrounding the sphere may be gelatin or soft tissue, instead of fluid. This will change the expression of $\pi a^2 \overline{Y} \langle E_0 \rangle$, which results in a dynamic radiation force of different amplitude. But the main conclusion remains valid: the dynamic radiation force is of frequency $\omega_1 - \omega_2$, and its amplitude remains constant for all oscillating frequencies $(\omega_1 - \omega_2)$ that are small. There are a few cases where the value of radiation force function Y is extremely sensitive to the change of ka, for example, at the resonant frequencies of a gold sphere in water [Chivers *et al.*, 1982]. Under these conditions, $\overline{Y}$ in Eq. (4) may not be independent of $\omega_1$ and $\omega_2$. But in most of the situations, it is safe to assume the dynamic radiation force is of constant amplitude when we sweep the oscillating frequency $(\omega_1 - \omega_2)$.

## III. IMPEDANCE AND VELOCITY IN VISCOUS MEDIUM

From the theory developed above, the dynamic driving force on the sphere is known. To find out the oscillating speed of the sphere, we need to compute the resistance it will meet when the sphere tries to move back and forth. In this section, a method is

proposed to solve for the oscillating speed of the sphere, from which the acoustic emission from the sphere can also be derived.

**A. Impedance of the sphere**

For a rigid sphere oscillating back and forth at frequency $\Delta\omega$, the stress field around the sphere can be calculated. The net force on the sphere can be found by integrating the stress at the surface of the sphere. The radiation impedance of the sphere is equal to this force divided by the vibrating speed of the sphere, and represents the resistance the sphere will "feel" when pushing its surrounding medium back and forth. Oestreicher (1951) derived a radiation impedance formula for a rigid sphere oscillating in a viscoelastic medium

$$Z_r = -i\frac{4\pi a^3}{3}\rho\Delta\omega\frac{\left(1-\frac{3i}{ah}-\frac{3}{a^2h^2}\right)-2\left(\frac{i}{ah}+\frac{1}{a^2h^2}\right)\left(3-\frac{a^2k^2}{aki+1}\right)}{\left(\frac{i}{ah}+\frac{1}{a^2h^2}\right)\frac{a^2k^2}{aki+1}+\left(2-\frac{a^2k^2}{aki+1}\right)}. \qquad (5)$$

Here, $k=\sqrt{\rho\Delta\omega^2/(2\mu+\lambda)}$, $h=\sqrt{\rho\Delta\omega^2/\mu}$, $\mu=\mu_1+i\Delta\omega\mu_2$, $\lambda=\lambda_1+i\Delta\omega\lambda_2$. Thus, the radiation impedance is determined by the radius of the sphere a, the density of the medium $\rho$, the shear elasticity and viscosity of the medium $\mu_1$ & $\mu_2$, and the volume elasticity and viscosity of the medium $\lambda_1$ & $\lambda_2$.

Another resistant force for vibration is the inertia of the sphere. For a sphere of mass m and oscillating velocity $Ve^{i\Delta\omega t}$, the force required to overcome the inertia of the

sphere is $F = m\frac{d(Ve^{i\Delta\omega t})}{dt} = im\Delta\omega Ve^{i\Delta\omega t}$. We define the "mechanical" impedance of the sphere as

$$Z_m = \frac{-F}{Ve^{i\Delta\omega t}} = -im\Delta\omega. \tag{6}$$

The total impedance of the sphere is the summation of these two: $Z = Z_r + Z_m$.

**B. Oscillating speed of the sphere**

The dynamic radiation force drives the sphere to vibrate, and the impedance represents the resistance towards vibration. Dividing the driving force by the impedance yields the vibrating speed

$$V = \frac{\langle F_d \rangle}{Z_r + Z_m}. \tag{7}$$

Once the vibrating speed of the sphere is known, the acoustic emission from the sphere can be calculated. The relevant formula can be found in the literature [Morse 1981]. But our experiments are done in a water tank. The reverberation of low frequency sound in the tank makes it difficult to measure acoustic emission from the sphere accurately. So experiments are designed to measure directly the vibrating velocity of the sphere instead.

Next, we show how the impedance of the sphere can change the vibration of the sphere. Some analysis of Eq. (5) will help us understand its influence on the vibration amplitude of the sphere. For soft tissues, $\lambda_1 \sim 10^9 Pa$, $\lambda_2 \sim 0 Pa \cdot s$; $\mu_1 \sim 10^4 Pa$, $\mu_2 \sim 10^{-2} Pa \cdot s$ [Frizzell *et al.*, 1977; Oestreicher 1951]. Therefore, ak is very small for

the a and $\Delta\omega$ considered (a<1mm and $\Delta\omega \le 2\pi \cdot 10^3$ rad/s), and can be safely set to zero. After this step, $Z_r$ is only a function of ah. Then, in the limit of large ah, one obtains purely mass impedance, and purely elastic impedance in the other limit. The absolute value of the total impedance $Z_r + Z_m$ is minimum at some resonance frequency approximately defined by the cancellation of total mass and elasticity contribution. Since the amplitude of the dynamic radiation force that is driving the sphere is constant for all vibrating frequencies, the vibration amplitude of the sphere will demonstrate a pattern of resonance versus vibration frequency. And the resonant pattern is determined by $\mu_1$ & $\mu_2$, not by $\lambda_1$ & $\lambda_2$. Computer simulation can demonstrate the result of this analysis more clearly. In the following simulations, it is assumed that the amplitude of the driving force on the sphere is $10^{-5}$ Newton.

Figure 2 is a simulation of the vibration speed versus vibrating frequency for a sphere in gel. The radius of the sphere is 0.59 mm. The gel has a shear elasticity of 4 kPa and its shear viscosity is set at 0, 1, 3, and 10 $Pa \cdot s$, respectively. The velocity shows a resonance near 400 Hz. Low viscosity corresponds to high resonant peak.

In Fig. 3, the value of shear viscosity is fixed and $\mu_1$ is changed over 30 fold. When $\mu_1$ decreases, the resonant peak is shifted to lower frequency and its magnitude becomes higher. In water, the peak is at frequency 0. In that case, only the right side of the peak is visible. The shear properties of the gel can change the velocity profile dramatically. Thus, it is possible to determine $\mu_1$ and $\mu_2$ by fitting the measured velocity profile. The size of the sphere can also alter the velocity profile. Smaller spheres have higher resonant peaks.

## IV. COMPARISON OF THEORY AND EXPERIMENT

### A. Experimental setup

The setup of experiment is shown in Fig. 4. The transducer vibrates the sphere back and forth at frequency $\Delta f$ in the water tank. A laser vibrometer is used to measure the vibration speed of the sphere. The laser vibrometer reflects a beam of light from the sphere, and detects the vibrating velocity of the sphere based on the Doppler frequency shift of the reflected laser. Four stainless steel spheres (provided by New England Miniature Ball LLC, Norfolk, CT) of different size are used in the experiment. We put each sphere at the focal spot of the transducer and measure its vibration response verses frequency $\Delta f$. The laser beams of the vibrometer and the laser pointer are used to align each sphere to make sure that they stay at the focal spot of the transducer. For experiments of spheres vibrated in water, each sphere is suspended with very fine suture material in a bifilar arrangement. For experiments of spheres vibrated in gel, the spheres are embedded in a gel phantom of $8\times10\times12cm^3$. The gel phantom is made from Bloom 300 gelatin powder (Sigma-Aldrich), with a concentration of 10%.

### B. Results in water

In water $\mu_1, \mu_2 \to 0$, resulting in $ah \to \infty$, the impedance formula of the sphere can be simplified and the velocity is

$$V \approx \frac{i3\langle F_d \rangle}{4\pi^2 a^3 (2\rho_s + \rho_0)} \times \frac{1}{\Delta f}. \tag{8}$$

Therefore, the amplitude of vibration velocity of the sphere should be inversely proportional to its vibrating frequency. Figure 5 is one set of measurement data for stainless steel spheres vibrated in water. Since the velocity is inversely proportional to frequency, the curve should be a straight line within the log-log scale in this plot. All the dots are measurement points and each solid line is fitted by the LMS method according to Eq. (8). The experimental data agree with the theory. Notice from Eq. (8) that one can also derive the dynamic radiation force on the sphere by the measured velocity. Hence it is also possible to estimate the energy density of the incident ultrasound through Eq. (4), since $\overline{Y}$ can be deduced from computer simulation. This is similar to the idea of using a steady radiation force on a sphere as a primary method for determining the intensity of ultrasound fields [Hasegawa *et al.*, 1975; Dunn *et al.*, 1977].

**C. Results in gel**

In gel, $\mu_1$ and $\mu_2$ are no longer negligible. The formula of the vibrating speed of the sphere becomes more complicated. Computer simulations based on Eq. (5) are used to fit the data of measurement. Figure 6 is a set of measurement data for four spheres of different size in one single gel phantom. The dots are measurement points. By adjusting the value of $\mu_1$, $\mu_2$ and a, four solid lines are fitted to the points. As expected, the measurements show a resonance of velocity versus frequency. For spheres of different size, the optimal value of $\mu_1$ lies within 3.9~4.2kPa and $\mu_2$ lies within 0.14~0.2 $Pa \cdot s$. In Fig. 6, all four solid lines are plotted assuming $\mu_1 = 4.1kPa$ and $\mu_2 = 0.17Pa \cdot s$, which come from averaging optimal parameters of four spheres. Independent measurement the gel's shear modulus $\mu_1$ by MRE [Muthupillai *et al.*, 1995] yields

2.5~4.5kPa, for shear wave frequency ranging from 200 Hz to 500 Hz. The same gel samples are also tested by the Dynamic Mechanical Analyzer (TA Instruments, New Castle, DE), and the measurement of $\mu_1$ is within 3.2~4.7kPa, for shear vibration frequency from 0 to 10 Hz. All three methods get similar results. We don't have a ground truth for $\mu_2$ of the gel. Our estimation is close to the typical values of $\mu_2$ for soft tissues estimated by Frizzell (1977).

## V. DISCUSSION

The method proposed in this paper is quite sensitive to changes of material properties. It is known that the shear modulus of gelatin phantom becomes larger as the gel ages. Experiments are repeated on the same gel phantom at different days. This method is able to detect a consistent increase of the stiffness of the gel every day. But the change of shear viscosity from day to day is not as prominent.

For a laser vibrometer to measure the velocity of an object, the medium around the target must be optically clear. This will limit the applications of this method. Ultrasound Doppler technique has been used to detect the motion of soft tissue under external mechanical vibration [Yamakoshi *et al.*, 1990]. We also have used ultrasound Doppler to measure the velocity of the vibrating sphere. The method employed to extract the velocity information is similar to that described in Yamakoshi's paper. The frequency of the Doppler transducer is 5 MHz, while the frequency of the ultrasound that used to vibrate the sphere is about 1 MHz. The received RF signal from the Doppler transducer is filtered out of the 1 MHz component before demodulation. Figure 7 shows the velocity profiles of a sphere in gel measured by laser vibrometer and the ultrasound Doppler

technique simultaneously. Both methods obtain similar results. Therefore, ultrasound can be used to measure the vibration instead of laser.

The model can produce good quantitative results for calibrated targets, like the sphere used in the experiments. For targets of different shapes, the concept presented in this paper is still valid. However, we need to calculate the radiation impedance of that particular target. Resonance of vibrating amplitude versus frequency is still expected, and can be used to deduce mechanical properties of the medium around that target. To apply this model to breast calcification, some practical issues need to be addressed. First of all, the target here is not exactly spherical, though many appear to be so. More work needs to be done to access the impact of violating the spherical assumption: how sensitive is the radiation impedance to the shape of the target, and how spherical are breast calcifications. Given the high contrast between normal fat tissues and malignant breast tissues, it is likely that assessment of the tissue properties around the microcalcification using this method can help in screening of breast lesion. Another practical issue is estimating the size of the calcification. Currently we are developing a system that combines x-ray mammogram and vibro-acoustography. The size of the calcification may be estimated by x-ray mammogram. Then this information may be used to estimate tissue properties by our method.

Another assumption of this model is that the medium surrounding the sphere is homogenous. In some applications, this assumption might be true only within a small distance from the target. The particle displacement in the medium surrounding the sphere can be estimated, using Eq. (15) in Oestreicher's paper (1951). This displacement field diminishes as the distance from the sphere increases. For a 0.5 mm radius sphere vibrated

in the gel phantom considered in Fig. 6, the particle displacement at 5 mm from the sphere is estimated to be about 5% of that at the surface of the sphere. Thus inhomogeneity beyond 5 mm from the sphere probably can be tolerated. This "effective" distance becomes larger, as the medium's shear elasticity and the sphere's radius increase. This knowledge will indicate the confidence we can have for the material property estimation.

## IV. SUMMARY

In this paper, we present a quantitative model for a sphere vibrated by two beams of ultrasound in a homogeneous viscous medium. The acoustic radiation force on the sphere is shown to have an oscillating component at the beat frequency of the two ultrasound beams. This driving force is used with the radiation impedance and the mechanical impedance of the sphere to calculate the vibrating velocity of the sphere. The theory predicts a distinct velocity profile (vibrating speed versus vibration frequency) for a sphere in water and in gel, which is confirmed by experiment. This method can be used to determine the local material properties of the medium surrounding the sphere. It can also be used to calibrate oscillatory driving force on the sphere.

## ACKNOWLEDGMENTS

The authors are grateful to Randall Kinnick for his experimental support. This work was supported by the Army Medical Research and Material Command under Grant DAMAD 17-98-1-8121, and by Grant HL61451 from National Institutes of Health.

## REFERENCES

Dunn, F., Averbuch, A. J., and O'Brien, W. D. (**1977**)"A primary method for the determination of ultrasonic intensity with the elastic sphere radiometer," Acustica **38:**58-61.

Chivers, R. C., and Ansosn, L. W. (**1982**)"Calculations of the backscattering and radiation force functions of spherical targets for use in ultrasonic beams assessment," Ultrasonics **20**:25-34.

Fatemi, M., and Greenleaf, J. F. (**1998**) "Ultrasound-stimulated vibro-acoustic spectrography," Science **280**:82-85.

Fatemi, M., and Greenleaf, J. F. (**1999**) "Vibro-acoustography: an imaging modality based on ultrasound-stimulated acoustic emission," Proc. Natl. Acad. Sci. USA **96**:6603-6608.

Fatemi, M., Rambod, E., Gharib, M., and Greenleaf, J. F. (**Jul. 2000**) "Nondestructive testing of mechanical heart valves by vibro-acoustography," 7th International Congress on Sound and Vibration, Garmisch-Partenkirchen, Germany, July 4-7, 2000.

Fatemi, M., Wold, L. E., Alizad, A., and Greenleaf, J. F. (**2002**) "Vibro-acoustic tissue mammography," IEEE. Trans. Med. Imag. **21**(1): 1-8.

Gao, L., Parker, K. J., Lerner, R. M., and Levinson, S. F. (**1996**) "Imaging of the elastic properties of tissue-a review," Ultrason. Med. Biol. **22**:959-977.

Hasegawa, T., and Yosioka, K. (**1969**) "Acoustic-radiation force on a solid elastic sphere," J. Acoust. Soc. Am. **46**:1139-1143.

Hasegawa, T, and Yosioka, K. (**1975**) "Acoustic radiation force on fused silica spheres and intensity determination," J. Acoust. Soc. Am. **58**:581-585.

Krouskop, T. A., Wheeler, T. M., Kallel, F., Garra, B.S. (**1998**) "Elastic moduli of breast and prostate tissues under compression," Ultrasonic Imaging **20**:260-274.

Morse, P. (**1981**) "Vibration and Sound," Acoustical Society of America 318-319.

Muthupillai, R., Lomas, D. J., Rossman, P. J., Greenleaf, J. F., Manduca, A., and Ehman, R. L. (**1995**) "Magnetic resonance elastography by direct visualization of propagating acoustic strain waves," Sci. **269**:1854-1857.

Oestreicher, H. L. (**1951**) "Field and impedance of an oscillating sphere in a viscoelastic medium with an application to biophysics," J. Acoust. Soc. Am. **23**:707-714.

Sarvazyan, A. P., Redenko, O. V., Swanson, S. D., Fowlkers, J. B., and Emelianov, S. Y. (**1998**) "Shear wave elasticity imaging: a new ultrasonic technology of medical diagnostics," Ultrason. Med. Biol. **24**:1419-1435.

Yamakoshi, Y., Sato, J., and Sato, T. (**1990**) "Ultrasonic imaging of internal vibration of soft tissue under forced vibration," IEEE. Trans. UFFC. **37**(2): 45-53.

**FIGURE CAPTIONS**

Fig. 1. Diagram of vibro-acoustography imaging system. The confocal transducer has two elements: a center disk and an outer ring. The object is vibrating at the beat frequency $\Delta\omega$. Its emission is detected by the acoustic hydrophone to reconstruct the image.

Fig. 2. Simulation of a sphere vibrated in gel. Parameters for the sphere: radius=0.59 mm, density = $7667 kg/m^3$. The gel has a shear elasticity of 4 kPa, and shear viscosity of 0, 1, 3, and 10 $Pa \cdot s$, respectively.

Fig. 3. Simulation of a sphere vibrated in gel. Parameters for the sphere: radius=0.59 mm, density = $7667 kg/m^3$. The gel has a shear viscosity of 0.1 $Pa \cdot s$, and shear elasticity of 1, 3, 10, and 30 kPa, respectively.

Fig. 4. Diagram of experimental setup. The laser vibrometer detects the motion of the sphere, which is vibrated by the transducer at frequency $\Delta f$. The laser beams of the vibrometer and the laser pointer are used to position the sphere at the focal spot of the transducer. The water tank is covered with sound absorbing pads.

Fig. 5. Vibrations of 440C stainless steel spheres of different size in water. Radius of the spheres: 0.44, 0.59, 0.64, 0.85 mm. Dots are measurement points and the solid lines are fitted by LMS method.

Fig. 6. Vibrations of 440C stainless steel spheres of different size in gel. Radius of the spheres: 0.44, 0.59, 0.64, 0.85 mm. Dots are measurement points and the solid lines are fitted by LMS method.

Fig. 7. Velocity measurement of a sphere in gel by laser vibrometer (dash line) and ultrasound Doppler method (solid line). Radius of the sphere: 0.59 mm.

Estimation of the gel's shear modulus is 1384 Pa from laser method and 1374 Pa from ultrasound method.

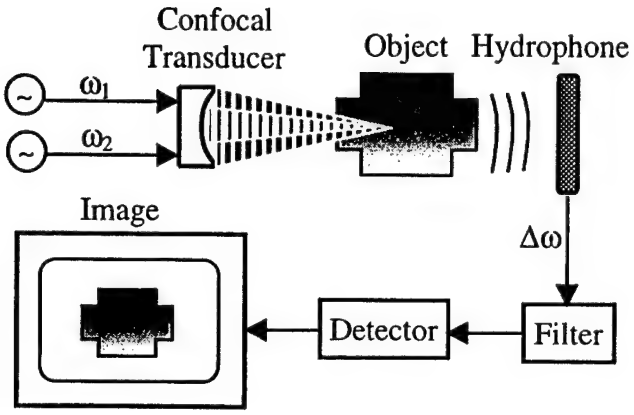



FIG. 1. Diagram of vibro-acoustography imaging system. The confocal transducer has two elements: a center disk and an outer ring. The object is vibrating at the beat frequency $\Delta\omega$. Its emission is detected by the acoustic hydrophone to reconstruct the image.

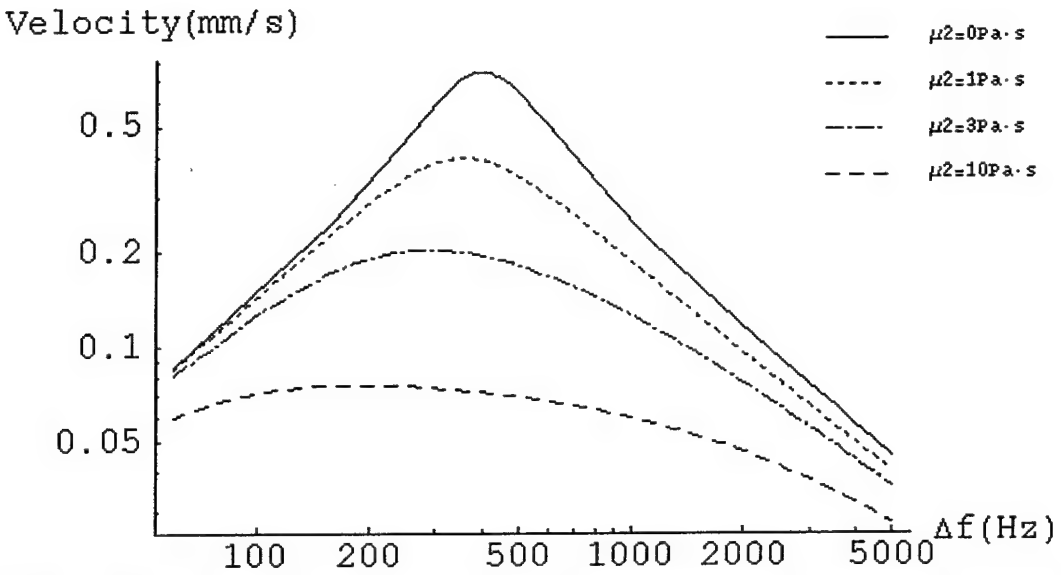



FIG. 2. Simulation of a sphere vibrated in gel. Parameters for the sphere: radius=0.59 mm, density= $7667 kg/m^3$. The gel has a shear elasticity of 4 kPa, and shear viscosity of 0, 1, 3, and 10 $Pa \cdot s$, respectively.

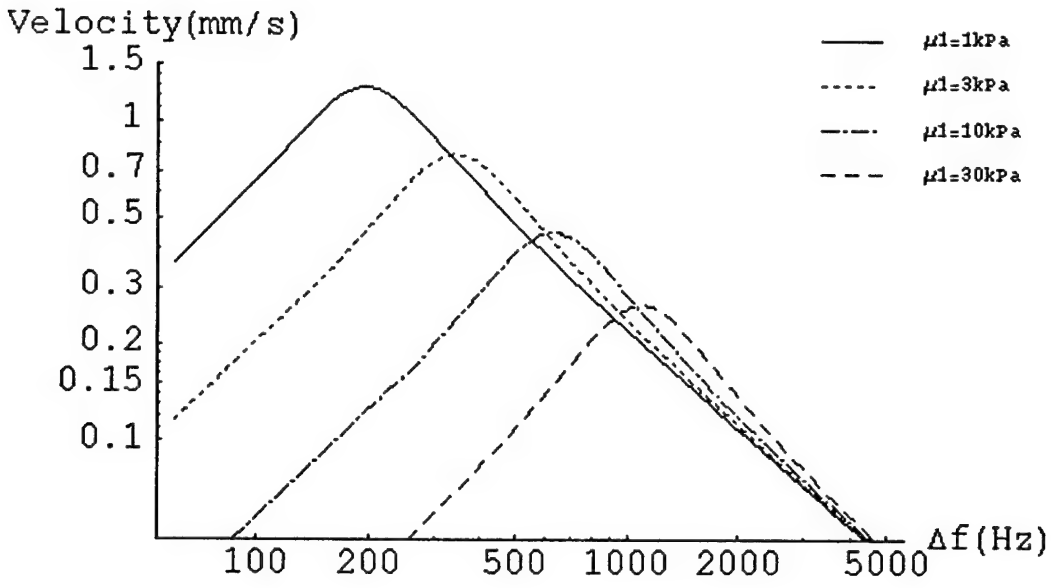



FIG. 3. Simulation of a sphere vibrated in gel. Parameters for the sphere: radius=0.59 mm, density= $7667 kg/m^3$. The gel has a shear viscosity of 0.1 $Pa \cdot s$, and shear elasticity of 1, 3, 10, and 30 kPa, respectively.

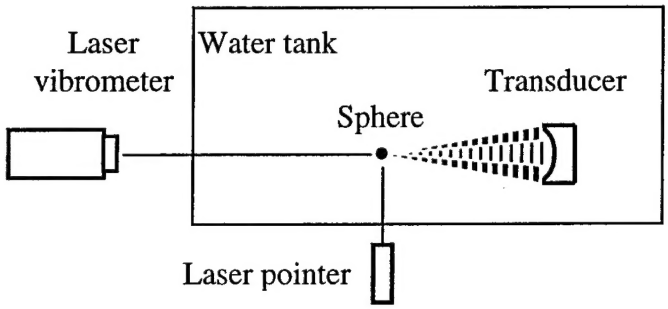



FIG. 4. Diagram of experimental setup. The laser vibrometer detects the motion of the sphere, which is vibrated by the transducer at frequency $\Delta f$. The laser beams of the vibrometer and the laser pointer are used to position the sphere at the focal spot of the transducer. The water tank is covered with sound absorbing pads.

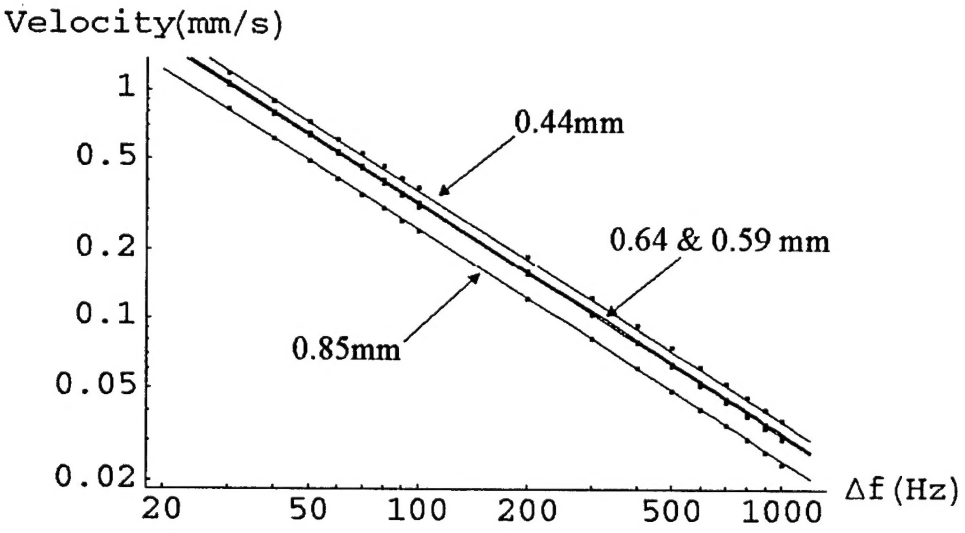



FIG. 5. Vibrations of 440C stainless steel spheres of different size in water. Radius of the spheres: 0.44, 0.59, 0.64, 0.85 mm. Dots are measurement points and the solid lines are fitted by LMS method.

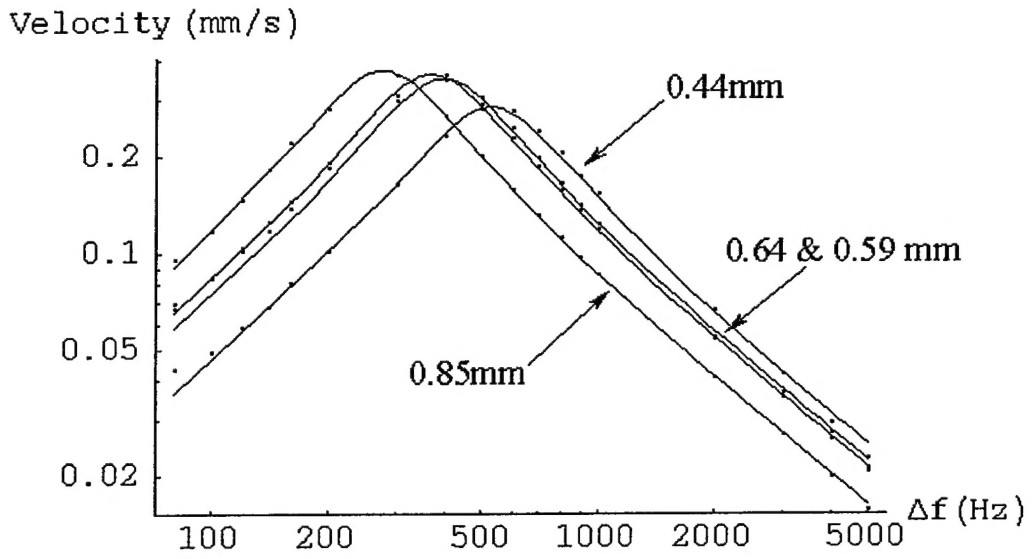



FIG. 6. Vibrations of 440C stainless steel spheres of different size in gel. Radius of the spheres: 0.44, 0.59, 0.64, 0.85 mm. Dots are measurement points and the solid lines are fitted by LMS method.

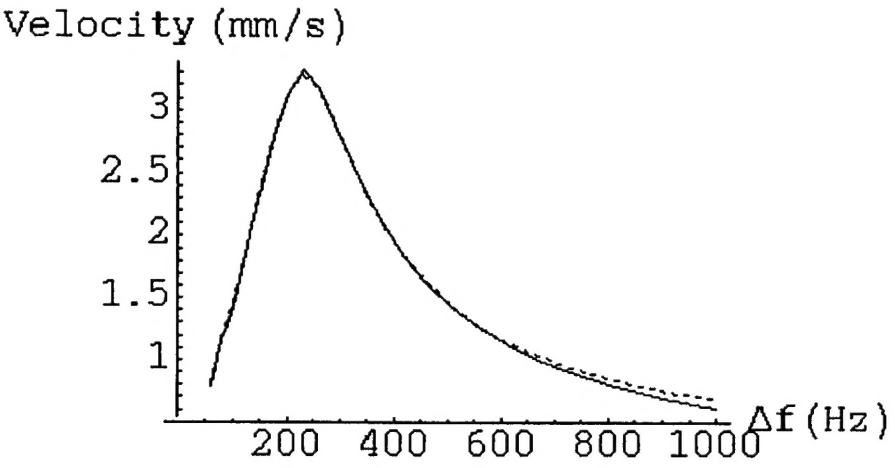



FIG. 7. Velocity measurement of a sphere in gel by laser vibrometer (dash line) and ultrasound Doppler method (solid line). Radius of the sphere: 0.59 mm. Estimation of the gel's shear modulus is 1384 Pa from laser method and 1374 Pa from ultrasound method.